Numerical Solution of Systems with Stochastic Uncertainties

—

A General Purpose Framework for Stochastic Finite Elements

Vom Fachbereich für Mathematik und Informatik
der Technischen Universität Braunschweig
genehmigte Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von
Dipl.-Math. Andreas Keese

Datum der Promotion:	6. April 2004
1. Referent:	Prof. Hermann G. Matthies, Ph. D.
2. Referent:	Prof. Dr. Rainer Hempel
3. Referent:	Prof. Dr. Christian Bucher
Eingereicht am:	18. Dezember 2003

Kurzfassung

Die vorliegende Arbeit entwickelt Verfahren zur numerischen Simulation von Systemen mit stochastischen Parametern, welche durch stochastische partielle Differentialgleichungen (SPDGLn) beschrieben werden. Nach einer Behandlung der Theorie linearer und nichtlinearer elliptischer SPDGLn werden Diskretisierungsverfahren für diese beschrieben. Während für die räumliche Diskretisierung eine existierende Simulationssoftware verwendet wird, erfolgt die stochastische Diskretisierung entweder durch die direkte numerische Integration von Statistiken unter Verwendung von Monte Carlo und Smolyak Quadraturverfahren oder durch eine Reihenentwicklung der Systemantwort in Tensorprodukten finiter Elemente und stochastischer Ansatzfunktionen. Die Koeffizienten in der Reihenentwicklung werden dabei entweder durch orthogonale Projektionen oder durch Galerkinverfahren in den stochastischen Dimensionen bestimmt.

Stochastische Galerkinvervahren sind der Hauptaugenmerk dieser Arbeit. Bei ihrer Anwendung entstehen große Systeme gekoppelter Blockgleichungssysteme, welche hier durch iterative Verfahren gelöst werden. Zur Lösung linearer SPDGln werden effiziente Darstellungen der Gleichungssysteme und iterative Löser entwickelt. Aufgrund der Größe der entstehenden Gleichungssysteme wird ein paralleler Löser bereitgestellt. Zur Lösung nichtlinearer SPDGLn werden modifizierte Newtonverfahren und Quasi-Newtonverfahren eingesetzt. Die adaptive Verfeinerung der Lösung wird durch ein duales Verfahren ermöglicht.

Diese neu entwickelten numerischen Verfahren werden in einer Allzwecksoftware für stochastische finite Elemente implementiert, welche es erlaubt, in existierenden Simulationscodes stochastische Unsicherheiten zu quantifizieren und ihre Auswirkung auf die Systemantwort zu bestimmen.

Abstract

The present thesis develops numerical techniques for the simulation of systems with stochastic parameters, modelled by stochastic partial differential equations (SPDEs). After treating the theory of linear and nonlinear elliptic SPDEs, discretisation techniques are presented. The spatial discretisation is performed by existing simulation software and the stochastic discretisation is carried out by directly integrating statistics or by expanding the solution in tensor products of finite element shape functions times stochastic ansatz functions.

Monte Carlo and Smolyak integration techniques are employed for the direct integration of statistics, whereas the discretisation by series expansions is realised either by orthogonal projections or by Galerkin methods in the stochastic dimensions.

Stochastic Galerkin methods are the focus of this thesis. They yield large systems of coupled block equations. For the solution of linear SPDEs, efficient representations of the linear block equations are developed and utilised in iterative solvers. Due to the size of the resulting systems of equations, a parallel solver is supplied. The solution of nonlinear SPDEs is performed by approximate Newton methods and by quasi-Newton methods. An adaptive refinement of the stochastic ansatz-spaces is implemented by a goal-oriented approach based on the solution of dual problems.

The numerical techniques described here are implemented in a general purpose software for stochastic finite elements that allows to introduce stochastic uncertainties into existing simulation codes and that permits to propagate the input uncertainties to the system response.

Acknowledgements

The work on this thesis was performed during my time as a research and teaching assistant (“Wissenschaftlicher Mitarbeiter”) at the Institute of Scientific Computing of the Technical University Carolo-Wilhelmina at Brunswick.

Foremost, I would like to thank Professor Dr. Hermann G. Matthies for his support and for the opportunity to work on this thesis. Our numerous discussions, his excellent guidance and his valuable suggestions were decisive in completing the thesis.

Also, I would like to thank Professor Dr. Christian Bucher and Professor Dr. Rainer Hempel for agreeing to referee the thesis.

The John von Neumann Institute for Computing at the Jülich research centre provided me with computing time on their massively parallel CRAY T3E computers. This support is gratefully acknowledged.

Further, I would like to thank Professor Dr. Knut Petras who introduced me to Smolyak integration.

I also would like to thank M.Sc. Yongke Yu who implemented the interface to the ANSYS software discussed in the final chapter of this thesis.

While working on the thesis, I was a collegial member of the graduate college “Interaction of Fluids and Structures” at the Technical University Brunswick. The numerous mini-symposia and the interdisciplinary exchange and discussions have been a valuable experience. I would like to thank the professors of the graduate college for their support and for their suggestions, especially Professor Dr. Heinz Antes and Professor Dr.-Ing. Dieter Dinkler. I also would like to thank the other collegial members and scholarship holders for the enjoyable atmosphere and for our numerous discussions.

The support of my colleagues at the Institute of Scientific Computing was valuable in writing this thesis. For the cooperative and enjoyable working conditions, I would like to thank my colleagues (in chronologic order) Dr. habil. Jörg Weimar, Dr. Christian Heimann, Dr. Jan Steindorf, Dr. Josef Schüle, Dr. Marcus Meyer, Dipl.-Math. Christian Frick, Dipl.-Inf. Oliver Kayser-Herold, Dipl.-Inf. Markus Krosche, M.Sc. Denis Frolov, Dr.-Ing. Rainer Niekamp, M.Eng. Tarin Srisupattarawanit, Dipl.-Ing. Thomas Peter Fries, Dipl.-Phys. Elmar Zander, and Simone Fischer.

I would like to thank Britta for her support and encouragement. Also, I would like to thank my friends for their support; special thanks go to Olaf for the proofreading. Finally, I would like to thank my parents Susanne and Albert for providing me with a sound education and I would like to thank them and my brother Christian for their constant support.

Braunschweig, December 2003 Andreas Keese

Contents

Chapter 1

Introduction

Numerical simulations of engineering systems require mathematical models for the problems of interest. These models are usually affected by uncertainties, which are either caused by a lack of information (epistemic uncertainties) or by intrinsic variabilities of the system parameters (aleatoric uncertainties).

Due to the always remaining uncertainties it is usually not clear to what degree the prognoses of numerical simulations match with reality. Computing power and numerical techniques are ever improving. It is hence more and more likely that the quality of prognoses obtainable by numerical simulations is no longer limited by the discretisation error but by the uncertainties about the system. If prognoses are computed with accuracies higher than merited by the uncertainties, then part of the available computing power does not contribute to obtaining more reliable results and is wasted.

More reliable prognoses may be obtained by quantifying the input uncertainties and by propagating these input uncertainties to the response. If a stochastic model is used for the uncertainty quantification, then the system is modelled by a stochastic partial differential equation (SPDE) and the uncertainty propagation is computed by solving the SPDE.

This thesis treats the theory, the discretisation, and the numerical solution of SPDEs. It focuses on numerical techniques for the discretisation by stochastic finite element methods and presents a general purpose software framework implementing these techniques.

1.1 The Relation to Previous Works

Previous works and their relation to this thesis are discussed next. Following this, Section 1.2 gives an overview of this thesis. Note that parts of this thesis were already published [80–87, 109–113, 187, 188]; see the commented list on page 192.

1.1.1 Stochastic Models of Uncertainties

The simulation of systems with stochastic uncertainties is a fast growing area of research. Stochastic models were employed in several applications, for example, in the engineering sciences [e.g. 19, 38, 56, 58, 67, 94, 97, 169, 176], or in the earth sciences [e.g. 26, 33, 150, 158]. For comprehensive overviews of the field see the reviews [80, 107, 156, 169] and the references therein.

Uncertainties in physical quantities that vary in time or in space may be modelled by *stochastic processes* [e.g. 14, 36, 93, 133] or by *random fields* [e.g. 1, 26, 62, 63, 173, 174]. As this thesis focuses on randomness in space, the term *random fields* is used.

The modelling of system parameters by random fields requires information about their statistics, which is sometimes seen as a disadvantage [37] as these are hard to obtain. But even if few statistical data are available, it may be better to make ad-hoc assumptions than to ignore the uncertainties. Statistics of system parameters may be obtained by sampling [11, 26, 102, 150, 162, 164] or by simulating random microstructures in stochastic homogenisation techniques [73, 74, 78, 79, 164, 172]; see the review [80] for a brief discussion.

Whether stochastic models are valid for describing uncertainties may be answered either by philosophical reasoning [26, 37, 119] or by comparing their prognoses with reality [99]. Stochastic models are not the only approach to uncertainty quantification. Examples of other approaches are fuzzy set techniques [189] and fuzzy finite element techniques [118, 146], interval analysis [e.g. 3], its generalisations to ellipsoidal and convex modelling [e.g. 37], and anti-optimisation procedures [e.g. 37]. A related issue are intrinsic heterogeneities on microscopic length scales treated by homogenisation techniques [e.g. 172, 191].

1.1.2 Stochastic Partial Differential Equations

Describing uncertainties in a physical system by stochastic models results in the system response being a random field or a stochastic process. The governing mathematical model is a *stochastic partial differential equation* (SPDE).

This thesis is concerned with randomness in space and hence does not consider stochastic differential equations (SDEs) [e.g. 91, 129], nor does it explicitly treat

partial differential equations with white noise forcing [e.g. 93, 141, 152, 175].

Publications on *stochastic finite elements* (SFEM) [e.g. 12, 57, 80, 107, 156, 169, 181] mostly consider SPDEs with ordinary random fields as parameters. This thesis focuses on this case and only briefly discusses SPDEs with generalised random fields [15, 16, 26, 44, 71, 93, 171].

A variational theory for elliptic linear SPDEs with ordinary or generalised fields was given by Benth and Gjerde [15]. Theories for linear SPDEs with ordinary random fields were discussed by Babuška, Tempone, Deb, Zouaris, and Chatzipantelidis [8, 11, 12, 34]. Theories for linear and nonlinear SPDEs were given in [84, 113] and will be presented in Section 2.3.

This thesis neither considers the instationary case—for results on ordinary differential equations with stochastic coefficients, see Babuška and Liu [10]—nor does it consider uncertain spatial domains—see Babuška and Chleboun [9] for results on the latter case for a non-stochastic model of uncertainty.

1.1.3 The Discretisation of Stochastic Partial Differential Equations

The ultimate goal in solving an SPDE is the computation of response statistics, like the mean of the solution, its variance, or the probability that it exceeds some threshold. To compute such statistics it is necessary to discretise the SPDE both in the spatial and in the stochastic dimensions. The spatial discretisation is performed here by finite element techniques. This is combined with stochastic discretisation methods, resulting in a discretisation of the SPDE by stochastic finite element methods (SFEM).

The stochastic discretisation requires the evaluation of high-dimensional integrals. Which techniques are efficient for the quadrature depends on properties of the system, like the coefficient of variance or the correlation length of the system parameters, and on the statistics being computed.

Monte Carlo methods [e.g. 22] may be utilised for the high-dimensional integration, but they require a high computational effort and they are badly suited if small probabilities are computed or if high accuracies are required. Hence, various alternatives were devised. Quasi-Monte Carlo methods [22] may be more efficient than Monte Carlo methods and the first and second order reliability techniques FORM and SORM [e.g. 67, 116] may be used to compute small failure probabilities. Smolyak quadrature methods [163] will be used here and it will be seen that they may be an efficient alternative to Monte Carlo methods for solving SPDEs.

Techniques based on the direct integration of statistics require to solve many realisations of the SPDE. As an alternative, the solution may be expanded in a

series of tensor products of spatial times random functions. Once the coefficients of the series expansion are obtained, statistics may either be computed analytically or by sampling from the series expansion.

Examples of series expansion techniques are certain response surface methods [e.g. 18, 88], perturbation methods [e.g. 67, 90, 131], methods based on Neumann-series [e.g. 8, 57, 132], non-intrusive methods based on orthogonal projections [58, 84, 113] and Galerkin methods in the stochastic dimensions [12, 15, 57, 113, 181]. An overview of such techniques is given in the reviews [80, 107, 156, 169].

Here, series expansion techniques based on orthogonal projections and Galerkin methods in the stochastic dimensions will be used, where the focus will be on Galerkin methods. These were devised by Ghanem and Spanos who termed their method the spectral stochastic finite element-method. Since their influential work [57], stochastic Galerkin methods have been applied to various linear stationary problems [e.g. 46, 49, 51, 52, 54, 86, 110, 135] and to various linear instationary problems [47, 48, 55, 76, 96, 176, 183]. Here, they will be applied to stationary linear and nonlinear SPDEs.

1.1.4 Solvers and Numerical Techniques

The direct integration techniques require to solve many statistically independent realisations of the SPDE. Each realisation is a partial differential equation that may be solved by existing software, e.g. by a finite element software. This simulation software will be called the *deterministic code* or the *deterministic solver*.

While direct integration techniques compute the solution of the SPDE by solving many uncoupled systems of equations, stochastic Galerkin methods require solving large systems of coupled block-equations. For each stochastic ansatz-function, these block-equations contain one block of equations having the size of the spatial discretisation.

Numerical techniques for solving these block-equations were presented in [52, 135]. These were based on classical block-iterative methods and on expanding the block-system matrix in the stochastic ansatz. A criterion for choosing the expansion block-system matrix was given in [113] and is presented in Section 4.1. A more efficient representation of the block-matrix and additional iterative solvers were developed in [109–111]. The latter works are presented here in sections 4.1 and 4.2.

The resulting system of block-equations is very large. Hence, parallel solvers were developed. Earlier versions of the parallel solver for linear SPDEs discussed in Section 4.3 were published in [83, 85, 86].

Few works solving nonlinear SPDEs by series expansions were published so far. Non-intrusive solvers based on orthogonal projections were presented in [58]

and in [84, 113]. The latter works use Smolyak quadrature to evaluate the projections and are presented in Section 3.4.2. Solvers for Galerkin discretisations of nonlinear SPDEs were first published in [81, 87, 112, 113] and are discussed in Section 4.4. Here, the resulting nonlinear system of block-equations is solved by quasi-Newton methods and the deterministic solver is used to precondition the BFGS-solver. A problem in solving nonlinear SPDEs by Galerkin methods is the integration of the residual. This problem is solved here and in [87, 112] by employing Smolyak quadrature.

Various techniques for the à posteriori error estimation, for a refinement of the solution, or for the computation of sensitivities of SPDE solutions were proposed and implemented; see [53, 96] and in particular [136] and the references therein. These techniques usually employ heuristic arguments or compute derivatives of the answer with respect to independent random variables. A goal-oriented approach based on dual techniques [89, 101, 145] was published in [82] and is presented here in Section 4.5.

1.1.5 General Purpose Software Framework

Various general purpose implementations for the computation of failure probabilities exist; see e.g. [19] and the references therein or [17, 20, 21, 59, 98]. However, these do not support the stochastic Galerkin methods employed here.

Chapter 5 discusses the general purpose software for stochastic finite elements StoFEL ("Stochastic Finite Element Library"), which implements the numerical techniques of this thesis. A general purpose implementation is obtained by coupling to an existing finite element software for the spatial discretisation in a black-box fashion and by running the deterministic solver as a preconditioner. To show the generality of the approach and to allow the solution of general SPDEs, the commercial finite element software systems FEMLAB [31] and ANSYS [4] were integrated within the StoFEL software [187, 188]; see Section 5.2.

1.2 Thesis Outline

This thesis is structured as follows: The second chapter discusses the modelling and the theory of stochastic partial differential equations (SPDEs). These are introduced in Section 2.1 by a simple stochastic groundwater-flow example. The discussion of this example also gives a more detailed overview of the problems solved in this thesis.

Section 2.2 reviews the modelling of Gaussian and non-Gaussian random fields. Following this, theories for linear and nonlinear elliptic stochastic partial

differential equations are presented in Section 2.3. A brief review of the theory of linear SPDEs with generalised random fields is given in Section 2.3.2.

Discretisation techniques for random fields and for SPDEs are discussed in the third chapter. Random fields are discretised by the Karhunen–Loève expansion (see Section 3.1.1). This requires the numerical solution of eigenvalue problems (Section 3.1.2) and existing finite element (FE) software is employed for this. Explicit estimates for the errors introduced by this FE approximation are computed in Section 3.1.4. By discretising all random fields in the SPDE, a representation of the SPDE in a countable number of mutually independent random variables is obtained (see Section 3.1.5).

The discretisation of the SPDE is performed first in the spatial and then in the stochastic dimensions. The spatial discretisation (see Section 3.2) yields a semi-discretisation of the SPDE to which a stochastic discretisation technique is applied. Two types of techniques are employed for the stochastic discretisation: direct integration methods (see Section 3.3) and series expansions (see Section 3.4).

Direct integration methods compute statistics by numerical integration (see Section 3.3.2). They require to approximate the SPDE in a finite number of independent random variables and the stability of this perturbation is investigated in Section 3.3.1. The numerical integration is performed here by Monte Carlo methods and by Smolyak quadrature (see sections 3.3.3 and 3.3.4). Other techniques for the evaluation of high-dimensional integrals are discussed in the Appendix B.

Series expansion methods for SPDEs are discussed in Section 3.4. These expand the solution in a series of random fields. Here, a series expansion in tensor products of spatial times stochastic functions is used (see Section 3.4.1). Two series expansion discretisations are considered: One is based on orthogonal projections (Section 3.4.2) and the other is a Galerkin method (Section 3.4.3). Steps towards obtaining convergence rates are discussed in Section 3.4.4.

The Galerkin methods require to solve large systems of coupled block-equations. Solution techniques for this are dealt with in Chapter 4. For linear SPDEs, two efficient representations for the block-equations are developed and analysed in Section 4.1. These are used to implement iterative solvers for linear SPDEs (see Section 4.2). Here, block-versions of the classical iterative solvers, preconditioned Krylov subspace methods, and multilevel techniques are presented, where the preconditioning is performed by calling the deterministic solver in a black-box fashion. Due to the size of the resulting systems of equations, a parallel solver is supplied, which is used to solve linear SPDEs in several millions of unknowns; see Section 4.3.

Nonlinear SPDEs are also discretised here by stochastic Galerkin methods. Modified Newton methods and quasi-Newton methods are used to solve the resulting system of nonlinear block-equations (Section 4.4). A problem in doing so is the evaluation of the residual. It is found in Section 4.4.1 that Monte Carlo meth-

ods may be badly-suited for integrating the residual, whereas high-dimensional quadrature methods are found to be an efficient alternative.

An adaptive solver for linear SPDEs is implemented in Section 4.5, which utilises a goal-oriented approach based on the solution of dual problems.

All these numerical techniques are implemented in a general purpose software for stochastic finite elements (see Chapter 5). This software allows to introduce stochastic uncertainties into existing simulation codes by calling them in a black-box manner. As an example of its versatility, Section 5.2 presents the coupling with the commercial finite element codes FEMLAB [31] and ANSYS [4].

The main body of the thesis ends in Chapter 6 with conclusions and directions for further research. This is followed by some appendices: Basics of stochastic analysis and of polynomial chaos expansions are summarised in Appendix A. Techniques for the evaluation of high-dimensional integrals are reviewed in Appendix B. A code example for the software framework is shown in Appendix C.

Notational conventions and used symbols are summarised in Appendix D. An alphabetical index is given on page 170, just before the bibliography. The thesis concludes with a commented list of publications written in its course; see page 192.

Chapter 2

Stochastic Partial Differential Equations and Random Fields

To introduce stochastic partial differential equations (SPDEs), Section 2.1 discusses a simple stochastic groundwater-flow example. Following this, random fields are introduced in Section 2.2. The theory of stochastic partial differential equations is presented in Section 2.3.

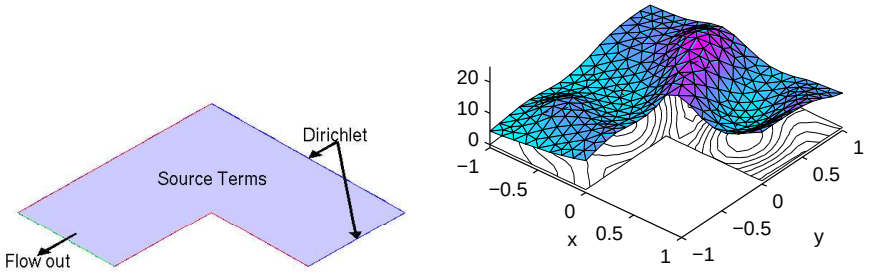
2.1 An Example for Modelling and Solving a Stochastic Partial Differential Equation

To give an impression of the problems aimed at in the present thesis, a simple example is discussed that exhibits all features of interest.

Partial Differential Equation: Consider a simple stationary groundwater flow example on the two-dimensional domain $R \subset \mathbb{R}^2$ shown in Fig. 2.1(a), where at first all parameters are deterministic. If the fluxes are related to the hydraulic head gradients by Darcy's law, then the hydraulic head $u(x)$ satisfies the elliptic partial differential equation (PDE) [68]

$$\begin{aligned} -\nabla \cdot (\kappa(x)\nabla u(x)) &= f_R(x), && x \in R, \\ n(x) \cdot (\kappa(x)\nabla u(x)) &= f_N(x), && x \in B_N, \quad B_N \cup B_D = \partial R, \\ u(x) &= f_D(x), && x \in B_D. \end{aligned} \tag{2.1}$$

Here, $\kappa(x)$ is the hydraulic conductivity and $f_R(x)$ accounts for the sources and sinks. The terms f_N and f_D are prescribed flows and hydraulic heads on the Neumann and Dirichlet boundaries B_D and B_N. The unit normal on B_N is $n(x)$. An outflow is prescribed on the green boundary in Fig. 2.1(a), no-flow conditions on



(a) The domain $R \subset \mathbb{R}^d$. (b) Realisation of $\kappa(x,\omega)$

Figure 2.1: Domain of groundwater flow example, realisation of conductivity.

the blue borders, a fixed hydraulic head on the red borders, and sources in all of the domain.

It is assumed here that a software—for example a finite element code—is available for solving the PDE of interest. This software will be called the *deterministic code* or the *deterministic solver*.

Stochastic Partial Differential Equation: It is difficult to measure soil parameters for real-world problems. Hence, they are often described by random models in the earth sciences [e.g. 26, 33, 48, 131, 158, 176].

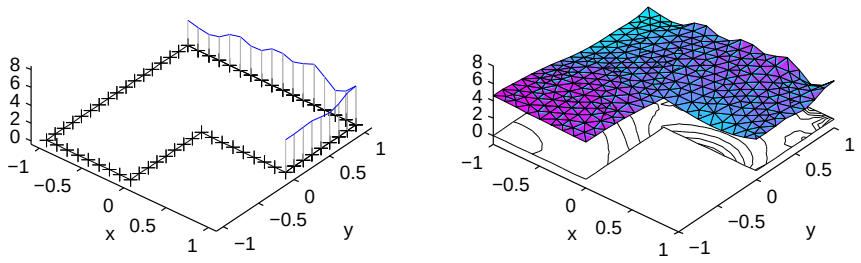
Let us demonstrate this for the hydraulic conductivity (see Appendix A.1 for some basics of stochastics): A random model is obtained by defining $\kappa(x)$ for each $x \in R$ to be a random variable $\kappa(x)\colon \Omega \to \mathbb{R}$ on a suitable probability space $(\Omega, \mathcal{B}, P)$. Consequently, $\kappa : R \times \Omega \to \mathbb{R}$ is a random field (see Section 2.2), where any elementary event $\omega \in \Omega$ gives a realisation $\kappa(\cdot,\omega)\colon R \to \mathbb{R}$ of the hydraulic conductivity.

An example of a realisation of the hydraulic conductivity used here is shown in Fig. 2.1(b). Uncertainties in the other parameters may also be modelled as random fields. In this example, the Dirichlet boundary conditions are also modelled as a random field; see the realisation in Fig. 2.2(a).

In combination with Eq. (2.1) the random system parameters yield the stochastic partial differential equation (SPDE)

$$\begin{aligned} -\nabla\cdot(\kappa(x,\omega)\nabla u(x,\omega)) &= f_R(x,\omega), && x \in R, \\ n(x)\cdot(\kappa(x,\omega)\nabla u(x,\omega)) &= f_N(x,\omega), && x \in B_N, \quad B_N \cup B_D = \partial R, \\ u(x,\omega) &= f_D(x,\omega), && x \in B_D, \end{aligned} \tag{2.2}$$

which will be understood in a variational sense. A theory of linear and nonlinear elliptic SPDEs is discussed in Section 2.3.



(a) Realisation of $f_D(x,\omega)$.

(b) Realisation of $u(x,\omega)$.

Figure 2.2: Realisation of f_D and of u

Although SPDEs will not be interpreted in a strong sense, a realisation-wise interpretation is instructive: Realisations of the random fields yield as realisations of the SPDE usual partial differential equations. These may be solved by the deterministic solver, yielding realisations of the solution of the SPDE. For example, Fig. 2.2(b) shows the realisation of the hydraulic head obtained for the realisations of κ and f_D shown in Fig. 2.1(b) and in Fig. 2.2(a).

The ultimate goal is to compute statistics of the solution. For example, properties of interest may be the expected hydraulic head $\mu_u(x) = \mathbf{E}\left(u(x,\cdot)\right)$, the variance $\mathrm{var}_u(x) = \mathbf{E}\left((u(x)-\mu_u(x))^2\right)$, or the probability that u exceeds some threshold, $p_u(x) = \mathrm{Prob}\{u(x) > u_0\} = \mathbf{E}\left(\chi_{(u_0,\infty)}(u(x))\right)$. All these statistics are integrals over Ω with respect to the probability measure and may be written as $s_u(x) := \mathbf{E}\left(s(u(x))\right) = \int_\Omega s(u(x,\omega))\, dP(\omega)$, where s is an appropriate function and where P is the probability measure. A brief overview of these basic facts is given in the Appendix A.1.

Spatial Discretisation: The numerical evaluation of such statistics requires a stochastic and a spatial discretisation. The spatial discretisation is performed here by employing the deterministic code. If the deterministic code is a finite element software, then the solution is expanded as $u(x,\omega) = \boldsymbol{N}(x)\boldsymbol{u}(\omega)$, where $\boldsymbol{N}(x) = (N_1(x),\ldots,N_n(x))$ is the vector of finite element shape functions and where $\boldsymbol{u}(\omega) = (u_1(\omega),\ldots,u_n(\omega))^T$ is the vector of random unknowns. As Section 3.2 will show, the spatial discretisation yields a set of linear equations with stochastic coefficients $\boldsymbol{K}(\omega)\boldsymbol{u}(\omega) = \boldsymbol{f}(\omega)$ to be solved for $\boldsymbol{u}(\omega)$. Here, $\boldsymbol{f}(\omega)$ is a random vector and $\boldsymbol{K}(\omega)$ is a usual stiffness matrix, but with random elements.

Direct Integration of Statistics: The statistics may be evaluated directly by numerical integration, for example by Monte Carlo simulations [e.g. 22]. A numerically tractable representation for this is obtained by expanding all parameters in a countable number of independent random variables $\theta = (\theta_1,\theta_2,\ldots)$ (see Sec-

tion 3.1). All random fields in the SPDE may then be written as functions of these random variables. For example, κ is then written as $\kappa(x,\omega) = \kappa(x,\theta(\omega))$, which is abbreviated as $\kappa = \kappa(x,\theta) = \kappa(x,(\theta_1,\theta_2,\ldots))$.

To make the numerical integration feasible, the SPDE needs to be approximated in a finite subset $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_m)$ of these basic random variables. This is a perturbation of the original SPDE and stability with respect to such perturbations is investigated in Section 3.3.1.

For example, the realisations of κ and of f_D shown in Fig. 2.1(b) and in Fig. 2.2(a) were computed by Karhunen–Loève expansions (see Section 3.1.1) in 24 and in 16 independent Gaussian random variables. The corresponding realisation of the solution shown in Fig. 2.2(b) was computed by approximating the SPDE in these $40 = 24+16$ random variables.

The approximation of the semi-discretisation in this finite number of random variables is $\boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{u}(\boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{\theta})$, where $\boldsymbol{u}(\boldsymbol{\theta})$ denotes the approximation of $\boldsymbol{u}(\boldsymbol{\theta})$ in the finite vector of random variables $\boldsymbol{\theta}$. Statistics may be evaluated as $s_u(x) \approx \mathbf{E}\,(s(u(x,\boldsymbol{\theta})))$. As the $\boldsymbol{\theta}$ are independent, the expectation may be written as

$$\mathbf{E}\,(s(u(x,\boldsymbol{\theta}))) = \int_{\Omega_1} \cdots \int_{\Omega_m} s(u(x,(\omega_1,\ldots,\omega_m)))\, dP_{\theta_1}(\omega_1)\cdots dP_{\theta_m}(\omega_m), \quad (2.3)$$

where $\Omega_i = \theta_i(\Omega)$ and where P_{θ_i} is the probability distribution of θ_i, $i = 1,\ldots,m$.

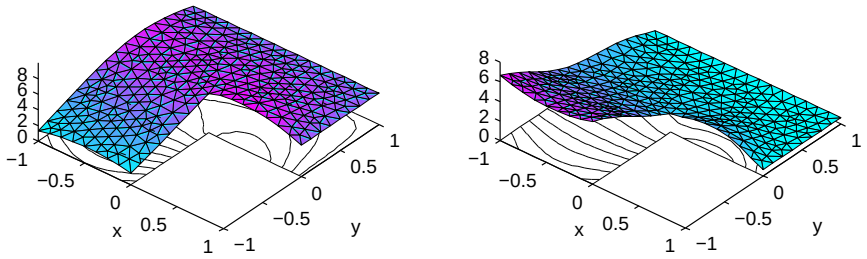
In principle, Eq. (2.3) may be evaluated by any integration technique, but the high dimensions make this difficult in practice. High-dimensional integrals are evaluated here by Monte Carlo integration [e.g. 22] and by Smolyak quadrature [163] (see Section 3.3 and Appendix B).

These integration techniques evaluate the integrand in Z integration points $\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(Z)} \in \Omega_1 \times \cdots \times \Omega_m$. Each integration point gives a realisation of the SPDE (a usual PDE), and each of these realisations is solved by calling the deterministic solver. Hence, Z uncoupled systems of equations need to be solved, each with the size of the spatial discretisation. The solutions give realisations of the integrand $s(u(x,\boldsymbol{\omega}^{(i)})), i = 1,\ldots,Z$, which are combined according to the integration rule to approximate the integral Eq. (2.3).

Series Expansions: The direct integration of statistics is expensive and hence various alternatives were proposed; see the reviews [80, 107, 156, 169]. This thesis focuses on discretisation techniques that expand the solution as a series

$$\boldsymbol{u}(\theta) = \sum_{\alpha} \boldsymbol{u}^{(\alpha)} H_\alpha(\theta) \quad \Rightarrow \quad u(x,\theta) = \sum_i \sum_\alpha N_i(x) H_\alpha(\theta) u_i^{(\alpha)}, \quad (2.4)$$

where the $H_\alpha(\theta)$ are appropriate functions in the basic independent random variables $\theta = (\theta_1,\theta_2,\ldots)$; see Section 3.4 for a discussion of these discretisation techniques.



Mean $\mu_u(x)$. Standard deviation $\sqrt{\mathrm{var}_u(x)}$

Figure 2.3: Mean and standard deviation of the hydraulic head.

In principle, any set of admissible linearly independent functions H_α may be used for the ansatz, but the high dimensions complicate this in practice. This thesis uses polynomial expansions in the stochastic dimensions. In particular, the H_α are chosen as orthogonal multivariate polynomials spanning the so-called polynomial chaos (see Appendix A.2.3).

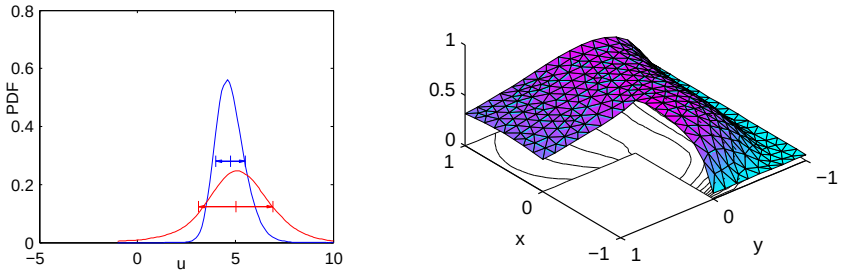
Together with the spatial discretisation, u is thus expanded in tensor-products of finite element shape functions times multivariate polynomials in the stochastic dimensions. Once all coefficients $u_i^{(\alpha)}$ are known, realisations of u may be generated at negligible costs and statistics of interest may be computed either analytically or by sampling from the response surface (see Example 2.1 on the next page).

Various techniques may be used to compute the coefficients $\boldsymbol{u}^{(\alpha)}$; see the reviews [80, 107, 156, 169]. This thesis concentrates on the solution by Galerkin methods in the stochastic dimensions (see Section 3.4.3). The application of Galerkin conditions yields the system of block equations

$$\mathbf{E}\left(\boldsymbol{K}(\theta)\sum_\alpha \boldsymbol{u}^{(\alpha)}H_\alpha(\theta)H_\beta(\theta)\right) = \mathbf{E}\left(\boldsymbol{f}(\theta)H_\beta(\theta)\right), \qquad \text{for all } \beta, \tag{2.5}$$

which is to be solved for all vectors $\boldsymbol{u}^{(\alpha)}$. In contrast to the direct integration in Eq. (2.3), the Galerkin method does not require to perturb the operator by explicitly approximating it in a finite number of independent random variables. Instead, the solution is obtained here as a projection and the representation in a finite number of random variables is obtained by choosing an appropriate stochastic ansatz.

Another difference to the direct integration methods is that these yield many uncoupled systems of equations, while the Galerkin method yields one coupled system of block-equations. The number of equations in this block-system is large and grows fast with the stochastic dimensions and with the degree of the stochastic ansatz.



(a) PDF $p_u(x)$

(b) The probability $P\{u(x) \geq 5\}$.

Figure 2.4: Statistics obtained by sampling the hydraulic head

Iterative solvers for the resulting block-equations will be discussed in Chapter 4, both for linear SPDEs and for nonlinear SPDEs. A general purpose software framework for the solution of SPDEs will be presented in Chapter 5.

Example 2.1: The SPDE Eq. (2.2) was solved by a stochastic Galerkin method in polynomial chaos of degree 2 in 40 independent random variables (861 stochastic ansatz functions). With a spatial discretisation in 326 degrees of freedom, the resulting system had approximately 280.000 equations. It was solved by the software framework discussed in Chapter 5 using the preconditioned conjugate gradients solver presented in Section 4.2.1 (the solution took approximately ninety minutes on a 2 GHz Pentium).

A realisation of the solution $u(x,\boldsymbol{\theta})$ that was obtained by sampling from the resulting expansion is shown in Fig. 2.2(b). The mean and standard-deviation were computed analytically from the resulting response surface and are shown in Fig. 2.3.

The series expansion was used to obtain 200.000 samples of the approximate hydraulic head. Kernel density estimation techniques [e.g. 102] were applied to the samples to compute the probability density function (PDF) of the approximate solution shown in Fig. 2.4(a). The red curve shows the PDF of $u(x,\boldsymbol{\theta})$ at $x = (0,0)$ and the blue curve shows the PDF at $x = (-1/2, 1/2)$. The respective means and standard deviations are shown as horizontal lines.

Additionally, the samples of the response surface were used to approximate the probability that the hydraulic head exceeds some threshold; see Fig. 2.4(b).

2.2 Random Fields

This section gives a brief overview of random fields and introduces the random field models used here.

For introductions to the theory of random fields see [1, 14, 36, 93, 129]; more practically oriented expositions are [26, 62, 63, 133, 173, 174]; for discussions of generalised random fields see [26, 44, 71, 93]. A more detailed overview of random fields and of their discretisation is also given in the reviews [80, 107, 156, 169] and the references therein.

2.2.1 Characterisations of Random Fields

A random field κ on a region $R \subset \mathbb{R}^d$ and on a probability space $(\Omega, \mathcal{B}, P)$ may be interpreted as a set of random variables indexed by $x \in R$ or as a function-valued random variable. In both interpretations, a random field is a measurable function

$$\kappa\colon R \times \Omega \longrightarrow \mathbb{R}.$$

Probabilistic Characterisation: The first interpretation regards a random field κ as a set of random variables $\kappa(x,\cdot)\colon \Omega \longrightarrow \mathbb{R}$ indexed by $x \in R$. It is defined by all its finite-dimensional ("fi-di") distribution functions $F_{x_1\ldots x_n}(\hat{x}_1,\ldots,\hat{x}_n) = P\{\kappa(x_1) \leq \hat{x}_1 \wedge \ldots \wedge \kappa(x_n) \leq \hat{x}_n\}$, with $x_1,\ldots,x_n \in R$ and $\hat{x}_1,\ldots,\hat{x}_n \in \mathbb{R}$; see [1, 36] for details. The probability space needs not to be specified explicitly as it may be constructed from the fi-di distributions under weak consistency conditions by Kolmogorov's extension theorem [e.g. 129, Theorem 2.1.5].

Measure Theoretic Characterisation: Alternatively, a random field κ may be interpreted as a random variable that has functions on a region $R \subset \mathbb{R}^d$ as values. Any elementary event ω yields a *realisation* $\kappa(\cdot,\omega)\colon R \longrightarrow \mathbb{R}$ (see Fig. 3.2). The set of elementary events Ω may hence be identified with the set of all possible realisations. Thus, one may identify the sample space with a function space $\Omega \subset \{\omega \,|\, \omega : R \to \mathbb{R}\}$ and defining κ amounts to specifying a probability measure P_κ on this function space Ω. This may in principle be done in conformance with given fi-di probability distributions [e.g. 129]. However, the interaction of the topological structure with the measure-space structure complicates this as it may not be possible to define the measure on the Borel-σ-algebra. For example, one cannot define a Gaussian measure on an infinite-dimensional Hilbert-space [e.g. 75, Example 1.25]. For nuclear spaces, the measure on the topological space may be defined using the Bochner-Minlos Theorem A.1 (see Appendix A.2.2).

2.2.2 Generalised Random Fields

A description by generalised random fields [44, 93] may be required for highly erratic random fields, e.g. for the white noise process [70], for soil properties [26], or for responses of elastic structures under white noise wind loads [93, 175].

Generalised Functions as Realisations: A generalised random field κ may be defined [44, 93] as a random variable that has generalised functions as realisations (e.g. tempered distributions). As above, the probability space may be identified with the space of realisations, and if the space is nuclear then the probability measure may be defined via the characteristic functional using the Bochner-Minlos Theorem A.1. Details and the construction of the white noise process are presented in the Appendix A.2.2.

Fields of Kondratiev Distributions: If a stochastic partial differential equations involves multiplicative noise, then it may be necessary to define products of generalised random fields. It is not obvious how to do this, and this problem was overcome by Holden et al. [71] by defining random fields as stochastic distributions (Hida- or Kondratiev-distributions), and interpreting products as Wick products (see Section 2.3.2); stochastic distributions are discussed in the Appendix A.3.

2.2.3 Gaussian Random Fields

A *Gaussian random field* $\gamma(x,\omega)$ is a random field, for which all fi-di distributions are jointly Gaussian. To define a Gaussian random field on a region $R \subset \mathbb{R}^d$, it is sufficient [e.g. 1, 62] to specify its second order statistics, i.e. its mean $\mu_\gamma(x)$ and its covariance function $\mathrm{cov}_\gamma(x,y)$ for $x,y \in R$ (see Appendix A.1 for details).

Gaussian random fields are used frequently to model system parameters. This is partly due to their convenient properties: Uncorrelated Gaussian random variables are independent, and linear combinations of Gaussian random variables are also Gaussian. Their use is also often justified from a theoretical point of view: Gaussian fields occur naturally due to the central limit theorem [14]. Further, there is often only second order statistical information available in applications [150], and then Gaussian random fields are the maximum entropy model [133].

However, Gaussian random variables have a positive probability of being negative and may hence be inappropriate for describing material properties. For example, the hydraulic conductivity in groundwater-flow problems needs to be positive. Modelling it as Gaussian may result in an ill-posed problem (see Section 2.3). Thus, non-Gaussian models are required for describing physical properties.

2.2.4 Non-Gaussian Random Fields

To define a random field, all its fi-di distributions must be specified. As this is not feasible for general non-Gaussian random fields, a particular random field model must be chosen in practice.

Most techniques of this thesis may be applied to general random fields. But to obtain a feasible representation, non-Gaussian random fields will be defined in the numerical experiments either as transformations of Gaussian fields (see Section 2.2.4.1) or as series expansions (see Section 2.2.4.2). For overviews of other random field models used in practice see [62, 63, 128, 144].

2.2.4.1 Transformations of Gaussian Random Fields

It is well-known [e.g. 133] that any standard Gaussian random variable $\mathcal{N}(0,1)$ may be mapped to a random variable with distribution function F_κ by the transformation $F_\kappa^{-1}(\mathrm{erf}(\mathcal{N}(0,1)))$, where erf is the Gaussian distribution function. If γ is a Gaussian random field with zero mean and with unit variance and if $F_{\kappa(x)}$, $x \in R$, are given marginal distribution functions, then the transformation

$$\kappa(x,\omega) := \phi(x,\gamma(x,\omega)) := F_{\kappa(x)}^{-1} \circ \mathrm{erf}(\gamma(x,\omega)), \tag{2.6}$$

defines a non-Gaussian random field with marginal distributions $F_\kappa(x), x \in R$.

Usually, the second order statistics of κ are given and $\mathrm{cov}_\gamma(x,y)$ must be chosen accordingly. This requires to compute the inverse of the relation between the correlation function $\rho_\kappa(x,y)$ of κ (see Appendix D.3) and cov_γ. Analytical formulas for various special cases are given in [62, 128].

A method to compute cov_γ for given combinations of given marginal distributions $F_{\kappa(x)}$ and given correlation functions $\rho_\kappa(x,y)$ is the so-called NORTA ("NORmal To Anything") method [24]: Note that $\rho_\kappa(x,y) = \Phi_{x,y}(\mathrm{cov}_\gamma(x,y))$ with a function $\Phi_{x,y}$ depending only on $F_{\kappa(x)}$ and on $F_{\kappa(y)}$. As γ is assumed to have unit variance, this is a map $\Phi_{x,y}\colon [0,1] \to [0,1]$. It is shown in [24] that the function $\Phi_{x,y}$ is nondecreasing and that it is continuous under mild conditions. Due to these properties, the inverse mapping $\mathrm{cov}_\gamma(x,y) = \Phi_{x,y}^{-1}(\rho_\kappa(x,y))$ may be computed by unconditionally convergent numerical methods [e.g. 143, 160].

Example 2.2: Here are some examples (with $c_1, c_2\colon R \to \mathbb{R}$).

- If $\kappa(x,\gamma(x,\omega)) = c_1(x)$, then κ degenerates to a deterministic field.
- If $\kappa(x,\gamma(x,\omega)) = \exp(c_1(x)\gamma(x,\omega))$, then κ is lognormally distributed.
- If $\phi(x,\gamma) = \mathrm{erf}(\gamma(x,\omega))$, then κ is uniformly distributed.
- If $\kappa(x,\gamma(x,\omega)) = \gamma(x,\omega)^2$, then κ is χ^2-distributed.

- If $\kappa(x,\gamma(x,\omega)) = c_1(x) + c_2(x)\arccos(\mathrm{erf}(\gamma(x,\omega)))$, then the marginal distributions of κ are $\beta(1/2,1/2)$-distributions.

2.2.4.2 Representation in non-Gaussian Random Variables

It is sometimes proposed [e.g. 8, 34, 97] to represent a non-Gaussian random field as a finite series expansion

$$\kappa(x,\omega) = \mu_\kappa(x) + \sum_{i=1}^{m} \kappa_i(x)\,\theta_i(\omega), \tag{2.7}$$

where the $\theta_i(\omega)$ are mutually independent (non-Gaussian) random variables and where the κ_i are appropriate functions, e.g. $\kappa_i \in L^2(R)$. This representation is a special case of the Karhunen–Loève expansion (see Section 3.1.1) and may be obtained from experimental data by a principal component analysis [11].

As the field is described in a finite number of independent random variables, the approximations discussed in Section 3.1 are not needed. As a drawback of this model, the marginal distributions of κ are in general unknown if the θ_i are non-Gaussian.

2.2.5 Correlation Models

The definition of a random field requires to specify its spatial correlation structure. At least, its second order statistics (its mean and covariance) need to be known. Gaussian random fields are completely defined by this, but the following also applies to non-Gaussian random fields.

The mean is chosen here as $\mu_\kappa \in L^\infty(R)$. The covariance function $\mathrm{cov}_\kappa : R \times R \to \mathbb{R}$ must be positive semi-definite and symmetric [e.g. 26, Ch. 3.1]. The positive semi-definiteness is not easy to check in general, but for weakly homogeneous random fields the covariance function is of the form $\mathrm{cov}_\kappa(x,y) = c(x-y)$ [1, 174] and then Bochner's theorem [e.g. 147, Theorem IX.9] may be used to check it for positive semi-definiteness.

To model real world phenomena by random fields, sufficient statistical information needs to be obtained experimentally. As this may be difficult [150, 162], experiments are usually supplemented by expert knowledge and theoretical reasoning [26]. Alternatively, statistical models may be computed by assuming a random microstructure and by computing effective stochastic material parameters by homogenisation techniques [74, 78, 79]. The review [80] discusses the modelling of random fields in more detail.

The covariance of a homogeneous field is often modelled [e.g. 26] as

$$\mathrm{cov}_\kappa(x,y) = c(r) = c(\Delta x^T \boldsymbol{G} \Delta x), \qquad \Delta x := y - x, \quad x,y \in \mathbb{R}^d \tag{2.8}$$

or as

$$\mathrm{cov}_\kappa(x,y) = c(r) = c(\sqrt{\Delta x^T \boldsymbol{G} \Delta x}). \tag{2.9}$$

Here, $\boldsymbol{G}$ is a non-negative matrix. Its eigenvectors denote the directions of anisotropies. If $\boldsymbol{G}$ is the identity matrix, then an *isotropic* field is obtained.

Most publications in the field of stochastic mechanics do not incorporate data from measurements. Instead, it is common practice to use the models Eqs. (2.9) and (2.8), where c is one of the functions discussed below. To adapt these models to real-world data, one may fit their parameters to measurements [178].

First order autoregression models: The exponential function

$$c(r) = \sigma^2 \exp\bigl(-r/\alpha\bigr), \quad r \geq 0 \tag{2.10}$$

is the covariance of a random process on $\mathbb{R}^1$ with correlation length $\alpha > 0$ and variance σ^2 satisfying the Langevin equation of Brownian motion [174, 178]. Using this function in Eq. (2.9) or Eq. (2.8) yields an admissible covariance function on a region $R \subset \mathbb{R}^d, d = 1,2,3$ [26]. This model is often used in the literature.

However, it was argued [178, 181] that it is difficult to visualise a physical process causing such a covariance in two or three spatial dimensions. The generalisation of the first-order autoregressive process driven by white noise to two dimensions was shown [178] to have the covariance function

$$c(r) = \sigma^2 \alpha^{-1} r K_1\bigl(r/\alpha\bigr), \tag{2.11}$$

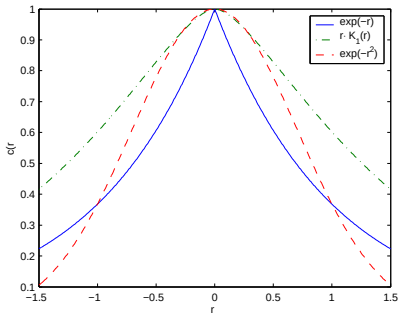
where K_1 is the modified first order Bessel function of the second kind.

Smooth Model: The "Gaussian" type covariance

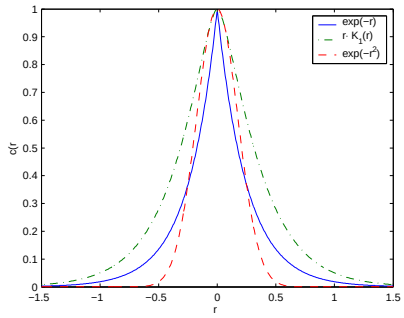
$$c(r) = \sigma^2 \exp\bigl(-\alpha^{-2} r^2\bigr), \qquad r \in \mathbb{R}, \tag{2.12}$$

is another frequently used model. The associated process is mean-square differentiable of any order. If this covariance is used with Eq. (2.9) for $R \subset \mathbb{R}^d$, then a valid covariance function is obtained for arbitrary dimensions d, as can directly be seen from Bochner's theorem [e.g. 147, Theorem IX.9].

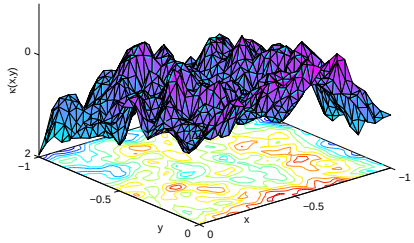
Example 2.3: Figures 2.5(a,b) show the above functions $c(r)$ for two correlation lengths. Figures 2.5(c–h) display representative realisations for centred Gaussian random fields on the rectangular domain $R = [0,1]^2$ with covariance functions $\mathrm{cov}_\kappa(x,y) = c(\|x-y\|) = c(r)$, where $c(r)$ is stated in the captions. The realisations were obtained by Karhunen–Loève expansions in 500 terms (see Section 3.1.1) discretised by triangular finite elements with piecewise linear shape functions in 1,148 degrees of freedom (see Section 3.1.2).



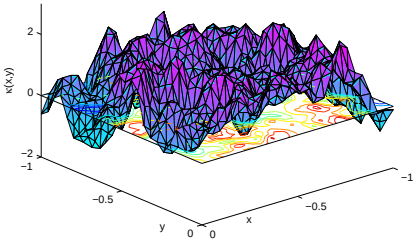
(a) Various $c(r)$ for $\alpha = 1$



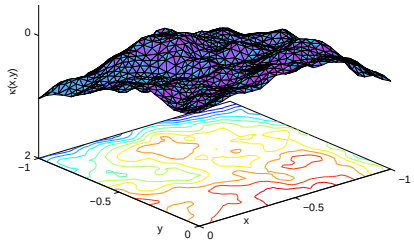
(b) Various $c(r)$ for $\alpha = 1/4$



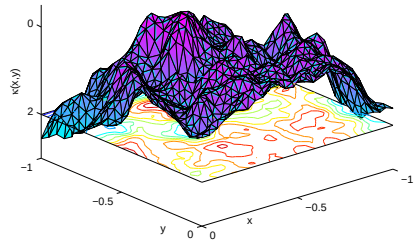
(c) Realisation, $c(r) = \exp(-r)$



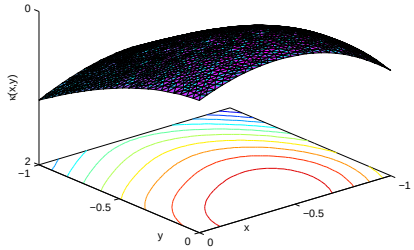
(d) Realisation, $c(r) = \exp(-r/4)$



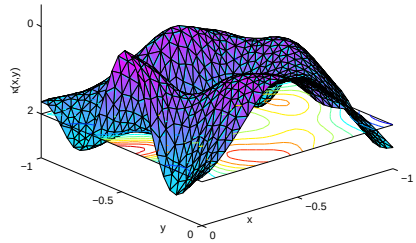
(e) Realisation, $c(r) = r \cdot K_1(r)$



(f) Realisation, $c(r) = r/4 \cdot K_1(r/4)$



(g) Realisation, $c(r) = \exp(-r^2)$



(h) Realisation, $c(r) = \exp(-r^2/4^2)$

Figure 2.5: Functions $c(r)$ and realisations of random fi elds.

The realisations demonstrate that the regularity of the realisations depends on the smoothness of $c(r)$ in $r = 0$: The covariance given by Eq. (2.10) is not differentiable in $r = 0$ and the according realisations 2.5(b,c) are continuous but nowhere differentiable [26]. The covariance Eq. (2.12) is smooth in $r = 0$, and the according realisations also look smooth. The plots demonstrate further that the correlation length controls the characteristic length of fluctuations (compare the plots on the left to the plots on the right).

2.3 Elliptic Stochastic Partial Differential Equations

This section discusses variational theories for linear and nonlinear elliptic SPDEs.

A variational theory for linear SPDEs with ordinary or generalised fields was given earlier by Benth and Gjerde [15] and by Besold [16]. A more detailed theory for linear SPDEs with ordinary random fields was presented later by Babuška, Tempone, Deb, Zouaris, and Chatzipantelidis [8, 11, 12, 34]. The theory for nonlinear elliptic SPDEs discussed here was published in [84, 113].

2.3.1 Linear Elliptic Stochastic Partial Differential Equations

In Section 2.1, it was shown how a stochastic partial differential equation (SPDE) is obtained by modelling the parameters of a partial differential equation (PDE) as random fields. The usual PDE theory [e.g. 27, 39, 126, 127] is now extended to linear elliptic stochastic partial differential equations.

The theory is presented here for a simple example, but it can be generalised to more general SPDEs that can be cast into variational form with coercive bounded bilinear form. The linear elliptic SPDE considered here is similar to the example used in Section 2.1. It is

$$\begin{aligned} -\nabla_x \cdot (\kappa(x,\omega)\nabla_x u(x,\omega)) &= f(x,\omega), \qquad && x \in R, \omega \in \Omega \\ u(x,\omega) &= 0, && x \in \partial R, \omega \in \Omega, \end{aligned} \tag{2.13}$$

where $R \subset \mathbb{R}^d$ is an admissible bounded set for the spatial domain with boundary ∂R and where ω denotes an elementary element from the probability space $(\Omega, \mathcal{B}, P)$. The operator ∇_x denotes the gradient with respect to x.

Eq. (2.13) is the strong form of the SPDE and as it is usual for PDEs, the SPDE will not be understood in this strong sense but in a variational sense. A strong interpretation would require Eq. (2.13) to hold for almost all $\omega \in \Omega$. This interpretation is only possible if all random fields in the SPDE are ordinary random fields. Then each realisation is a linear elliptic PDE, for which standard theory holds. If generalised random fields occur in the SPDE, then realisations

may be meaningless and then the strong form is inadequate. As for the deterministic case, the variational form of SPDEs is more general than the strong form and hence SPDEs will be interpreted here in a variational sense.

As a first step in generalising the theory of PDEs to SPDEs, the admissible vector space for u needs to be found. For the deterministic PDE analogon of Eq. (2.13), the solution would be in $V := \mathring{H}^1(R)$, where $\mathring{H}^1(R)$ is the completion of $C_c^\infty(R)$ in the Sobolev space $H^1(R)$. According to [8, 11, 12, 15, 16, 34, 171] it is natural to assume that the solution of the SPDE is an element of a tensor product space

$$u \in V \otimes (S) = \mathring{H}^1(R) \otimes (S) \cong \mathring{H}^1(R,(S)), \tag{2.14}$$

where (S) is an appropriate separable Hilbert space of random variables. The function space V on which the PDE analogon of the SPDE is defined on will be called the space of *admissible spatial functions*. The space of random variables (S) will be called the space of *admissible stochastic functions*.

The choice of (S) and hence the stochastic regularity of the solution depends on the stochastic regularity of f and of κ. Here, it will usually be assumed that $(S) = L^2(\Omega)$ and that this space is separable. According to [15, 16, 171], the following results still hold if (S) is a more general space, e.g. a separable space of stochastic distributions (see Section A.3).

More general spaces (S) may be required if $\kappa(x,\cdot)$, $x \in R$, is a stochastic distribution. But then special care must be taken to make the SPDE well-defined: If $u(x,\cdot)$ is also a stochastic distribution, then it is not obvious how the product $\kappa(x,\omega)\nabla_x u(x,\omega)$ [71, 171] may be defined; this will be discussed in Section 2.3.2.

For now, it is assumed that $\kappa \in L^\infty(R) \otimes L^\infty(\Omega)$ and that $(S) = L^2(\Omega)$. Hence, u and κ may be multiplied realisation-wise by the usual product.

Before introducing the variational form, let us define the extension ∇_x of the gradient operator to $V \otimes (S)$ (as usual, differentials are interpreted in a weak sense). For a single tensor product $u_1(x)u_2(\omega) \in V \otimes (S)$, let

$$\nabla_x : u_1(x)u_2(\omega) \mapsto (\nabla u_1(x))\, u_2(\omega). \tag{2.15}$$

By linearity and continuity, this may be extended to a linear bounded operator

$$\nabla_x : \mathring{H}^1(R) \otimes (S) \to L^2(R) \otimes (S). \tag{2.16}$$

For $v \in V \otimes (S)$, define $\ell(v) := \langle\langle f, v\rangle\rangle$, where $\langle\langle \cdot,\cdot \rangle\rangle$ is the duality paring between $(V \otimes (S))^* = V^* \otimes (S)^*$ and $V \otimes (S)$. With $\ell \in V^* \otimes (S)^*$, the variational formulation is to

$$\begin{aligned} &\text{find } u \in V \otimes (S), \text{ such that} \\ &a(u,v) = \ell(v), \qquad \text{for all } v \in V \otimes (S), \end{aligned} \tag{2.17}$$

where the bilinear form $a = a_\kappa$ is defined as

$$a(u,v) = a_\kappa(v,u) := \int_\Omega \int_R (\nabla_x v(x,\omega))^T \kappa(x,\omega) \nabla_x u(x,\omega)\, dx\, dP(\omega). \quad (2.18)$$

As extension of the deterministic case, the following result holds [8, 12, 15, 34]:

Theorem 2.1: *If $\kappa \in L^\infty(R) \otimes L^\infty(\Omega)$ with $\kappa(x,\omega) \geq \kappa_- > 0$ a.e. in $R \times \Omega$ and if $\ell \in (V \otimes (S))^*$, then the variational form Eq. (2.17) has a unique solution $u \in V \otimes (S)$. The solution depends continuously on ℓ in $(V \otimes (S))^*$.*

PROOF: Standard arguments show the continuity and coerciveness of $a_\kappa(v,u)$. The Lax-Milgram lemma [e.g. 27, 126] proves existence, uniqueness, and the continuous dependence on ℓ. □

The numerical algorithms will approximate f and κ by random fields $\hat{f}$ and $\hat{\kappa}$. Stability with respect to perturbations of κ in $L^\infty(R) \otimes L^\infty(\Omega)$ was shown in [12, Cor. 2.1]. However, the approximations of κ usually converge in $L^\infty(R) \otimes L^2(\Omega)$ and the following theorem shows stability with respect to such perturbations:

Theorem 2.2: *Let $f, \hat{f} \in L^2(R) \otimes L^2(\Omega)$ and let $\kappa, \hat{\kappa} \in L^\infty(R) \otimes L^\infty(\Omega)$, with both $\kappa(x,\omega), \hat{\kappa}(x,\omega) \geq \kappa_- > 0$ a.e. in $R \times \Omega$. Denote by $u, \hat{u} \in V \otimes L^2(\Omega)$ the solutions of $a_\kappa(u,v) = (f,v)_{L^2(R)\otimes L^2(\Omega)}$ and $a_{\hat{\kappa}}(\hat{u},v) = \left(\hat{f},v\right)_{L^2(R)\otimes L^2(\Omega)}$ for all $v \in V^* \otimes L^2(\Omega)$.*

Then there exists a constant $C > 0$, independent of $\hat{f}$ and $\hat{\kappa}$, such that

$$\|u - \hat{u}\|_{V\otimes L^2(\Omega)} \leq C \|f - \hat{f}\|_{L^2(R)\otimes L^2(\Omega)} + C \|\kappa - \hat{\kappa}\|_{L^\infty(R)\otimes L^2(\Omega)}.$$

PROOF: Note that $\kappa, \hat{\kappa} \in L^\infty(R) \otimes L^2(\Omega)$ as probability measures are finite. The continuous dependence on $\|f - \hat{f}\|$ follows from Theorem 2.1. To show the continuous dependence on κ, the second Strang lemma (see [165] or [27, p. 186]) may be employed as its proof relies only on the coercivity and on the boundedness of the bilinear forms. According to the second Strang lemma,

$$\|u - \hat{u}\|_{\tilde{V}} \leq C \left(\|f - \hat{f}\|_{L^2(R)\otimes L^2(\Omega)} + \sup_{w \in \tilde{V}, \|w\|=1} |a_{\hat{\kappa}}(u,w) - a_\kappa(u,w)| \right)$$

where $\tilde{V} := V \otimes L^2(\Omega)$. The second summand in the parantheses is equal to

$$\begin{aligned}
&\sup_{\|w\|=1} \int_\Omega \int_R \nabla_x u(x,\omega)^T (\hat{\kappa}(x,\omega) - \kappa(x,\omega)) \nabla_x w(x,\omega)\, dx\, d\omega \\
&\quad \leq \int_\Omega \sup_{x \in R} |\hat{\kappa}(x,\omega) - \kappa(x,\omega)| \sup_{\|w\|=1} \int_R \nabla_x u(x,\omega)^T \nabla_x w(x,\omega)\, dx\, d\omega \\
&\quad \leq C' \|\kappa - \hat{\kappa}\|_{L^\infty(R)\otimes L^2(\Omega)} \quad \text{by the Cauchy-Schwarz inequality in } L^2(\Omega).
\end{aligned}$$

□

The bilinear form a_κ defines a continuous, self-adjoint, positive definite operator $A_\kappa\colon V\otimes(S)\to V\otimes(S)^*$ with continuous inverse and the SPDE may be written as $A_\kappa u = f$. As both the operator and its inverse depend continuously on the data, the problem is well-posed.

Note that the boundedness of the material parameter away from zero is essential: It was shown in [8] that no solution of the linear SPDE Eq. (2.17) exists, if there is for every $\varepsilon > 0$ a region $R_\varepsilon \subset R$ with positive measure such that $P\{|\kappa(x,\omega)| < \varepsilon\} > 0$ for all $x \in R_\varepsilon$.

A large number of publications solving engineering problems described by elliptic SPDEs choose Gaussian random fields as material. These are not bounded away from zero and hence results of such publications must be considered with care. If Monte Carlo methods are used for the solution, then small or negative κ are usually rejected, but then the materials are not Gaussian. This rejection of small realisations is usually not possible if other discretisation methods are used, like perturbation methods or stochastic Galerkin schemes (see Chapter 3). It is thus problematical how one should interpret such results, even though the numerical results often match with the results obtained by Monte Carlo simulations.

The boundedness of κ from above is required for the application of the Lax-Milgram lemma to make A_κ bounded. For non-bounded κ, e.g. for lognormally distributed $\kappa - \mathbf{E}(\kappa)$, the variational formulation may be written as a minimisation problem [e.g. 27] and a solution may still be well-defined. However, the operator A_κ is in this case unbounded. Just as for deterministic PDEs, where this situation is frequently encountered (e.g. in electrostatic problems with the Coulomb potential), the unboundedness of the operator may be remedied by restricting it to an appropriate subspace of $\mathring{H}^1(R)\times(S)$, but this will not be discussed here further.

Note that the example for the SPDE Eq. (2.17) was kept simple for the sake of exposition; an extension to general linear SPDEs with bounded coercive bilinear form is straightforward.

2.3.2 Linear Stochastic Partial Differential Equations with Generalised Random Fields

It may be necessary to model system parameters as generalised random fields [26, 71, 93, 175]. If the right hand side is a generalised function, then it may be necessary to impose stronger regularity requirements on the material κ. More severe complications arise if the material parameter κ is a generalised random field. For instance, this may be necessary for the modelling of soil variabilities [26, 71].

If the material κ is a random field and if u is also a generalised random field, then it is not obvious how to interpret the product $\kappa(x,\omega)\nabla_x u(x,\omega)$. One possible

interpretation is to regard products between random fields as Wick products [71]. Alternatively, one may impose conditions on κ that make the usual product well-defined [16].

Wick SPDEs: Holden, Øksendal, Ubøe, and Zhang [71] model random fields as stochastic (Kondratiev) distributions and interpret products between generalised random fields as Wick products [179]. The Wick product will be denoted here as $\odot$ and the resulting SPDEs will be called *Wick SPDE*s. The Wick version of the SPDE Eq. (2.13) is

$$\begin{aligned} -\nabla_x \cdot (\kappa(x,\omega) \odot \nabla_x u(x,\omega)) &= f(x,\omega), & x &\in R \\ u(x,\omega) &= 0, & x &\in B_D. \end{aligned} \tag{2.19}$$

It may be solved by transforming it to a sequence of usual PDEs [71]. The strong formulation in [71] results in restrictions on the boundary conditions and source terms that may be relaxed by a variational formulation [108, 171].

Wick SPDEs have a different interpretation than SPDEs with pointwise multiplication of random fields, as $\mathbf{E}(u \odot v) = \mathbf{E}(u)\mathbf{E}(v)$ for (generalised) random variables u, v. As a consequence, for (e.g. linear) Wick SPDEs the mean of the solution is not influenced by higher statistical moments of the material parameters.

This behaviour does not agree with the results of homogenisation theory [e.g. 28, 120]. Neither does it agree with the usual interpretation of a stochastic system as a set of possible worlds across which statistics are taken. Of course, realisations cannot be obtained in the usual way if generalised random fields are involved, but even if SPDEs with ordinary random fields are interpreted in the Wick sense, the results do not agree with Monte Carlo simulations. The Wick product seems therefore not to be the right model for the problems aimed at here; see [71, pp.128ff.] for a comparison of Wick SPDEs and SPDEs with usual product.

A method for the numerical solution of Wick SPDEs by a stochastic Galerkin method (see Section 3.4.3) was presented by Theting [171]. The discretisation yields an upper block-triangular system of block equations. In contrast to this, Galerkin discretisations of SPDEs with usual product lead to coupled systems of block equations with less favourable structure. Hence, the numerical solution of Wick SPDEs requires less effort than the solution of SPDEs with usual product if the same ansatz space is used.

SPDEs with Generalised Random Fields and Normal Product: Benth and Gjerde [15] present a variational theory for linear SPDEs. Here, (S) in Eq. (2.14) may be a space of stochastic distributions, $(S) = (S)^{-\rho,-q}$ (see Section A.3). Assuming that the weak form associated with the SPDE is coercive and bounded on $H_0^1(R) \otimes (S)^{-\rho,-q}$, they apply the Lax-Milgram lemma to show the existence and uniqueness of solutions. Their results hold both for SPDEs with usual product and

for Wick SPDEs. The focus of their work are the convergence rates Eq. (A.21) for approximations in polynomial chaos.

How one can make the usual product between generalised random fields well-defined is investigated by Besold [16]. He shows that the pointwise multiplication may be well-defined if the generalised random field κ satisfies certain stochastic regularity requirements. In particular, it is required that $\kappa \in C^\infty(R) \otimes (\mathcal{E})^\infty$ with an appropriate space of stochastic distributions $(\mathcal{E})^\infty$ (see [16] for details). Imposing additional conditions guaranteeing coercivity and boundedness, Besold shows that a unique solution $u \in \mathring{H}^1(R) \otimes (S)^p$ of the elliptic SPDE exists, where $(S)^p$ is a Kondratiev distribution space (see [70, 71]).

The case $(S) = L^2(\Omega)$ is included in this theory. However, the requirement that κ is smooth in the spatial dimensions restricts the applicability somewhat. From the usual theory of PDEs, one would hope that one may choose $\kappa \in L^\infty(R) \otimes (\mathcal{E})^\infty$, but the theory in [16] cannot be extended to this case in a straightforward manner.

2.3.3 Nonlinear Stochastic Partial Differential Equations

The theory of nonlinear elliptic SPDEs devised in [84, 113] is presented here.

As an example for a nonlinear SPDE, it is assumed here that the material κ depends on the hydraulic head. For example, if the original SPDE models a groundwater problem, then the hydraulic conductivity might depend on the hydraulic head due to saturation effects. The resulting SPDE is then

$$\begin{aligned} -\nabla_x \cdot (\hat{\kappa}(x, u(x,\omega), \omega) \nabla_x u(x,\omega)) &= f(x,\omega), \qquad && x \in R, \\ u(x,\omega) &= 0, && x \in \partial R. \end{aligned} \tag{2.20}$$

The further developments may be applied to more general material laws, but to make the setting concrete, it is assumed here that

$$\hat{\kappa}(x, u(x,\omega), \omega) = \kappa(x,\omega) + c(x,\omega)\, u(x,\omega)^2, \tag{2.21}$$

where $\kappa(x,\omega)$ and $c(x,\omega)$ model the soil properties with $0 < \kappa_- \leq \kappa(x,\omega) < \kappa_+ < \infty$ and with $0 < c_- \leq c(x,\omega) < c_+ < \infty$ for almost every $x \in R, \omega \in \Omega$.

Just as for the linear case, solutions of the nonlinear SPDE will be sought here in tensor product spaces. Because of the nonlinearity, the spatial and the stochastic part of the solution are no more the Sobolev space $\mathring{H}^1(R)$ and the Hilbert space $L^2(\Omega)$, but they are the Sobolev space $V := \mathring{W}_p^1(R)$ and the Banach space $(S) := L^p(\Omega)$ with $p = 4$. The space for the solutions is $V \otimes (S) := V \otimes (S) \cong L^p(\Omega, V)$, and the right hand side is required to be an element of $V \otimes (S)^* \cong L^q(\Omega, V^*)$, where $V^* = W_q^{-1}(R)$, with $1/p + 1/q = 1$. As in the linear case, more general spaces may be used for (S).

The gradient operator ∇_x is defined analogous to Eqs. (2.15) and (2.16) as a linear and continuous map from $V \otimes (S)$ into $L_p(R, \mathbb{R}^d) \otimes (S)$. With the stochastic Nemicky operator

$$G\colon V \otimes (S) \to L_q(R, \mathbb{R}^d) \otimes (S)^* \tag{2.22}$$

$$G(u(x,\omega)) := (\kappa(x,\omega) + c(x,\omega)u(x,\omega)^2)\nabla_x u(x,\omega), \tag{2.23}$$

the semilinear form

$$a(u,v) := \int_\Omega \int_R \nabla_x v \cdot G(u)\, dx\, dP(\omega) \qquad u, v \in V \otimes (S) \tag{2.24}$$

defines a nonlinear, strictly monotone, hemicontinuous, coercive operator A from $V \otimes (S)$ into its dual. The variational form of the nonlinear SPDE requires to

$$\begin{aligned} &\text{find } u \in V \otimes (S), \text{ such that} \\ &a(u,v) =: \langle\langle A(u), v\rangle\rangle = \langle\langle f, v\rangle\rangle, \qquad \text{for all } v \in V \otimes (S), \end{aligned} \tag{2.25}$$

where $\langle\langle \cdot, \cdot \rangle\rangle$ is the duality paring between $V \otimes (S)$ and its dual. The same arguments as in the deterministic case [27, 39, 77] may be used to ascertain the existence and uniqueness of a solution. Summarising the above, we obtain the following result [84, 113]:

Theorem 2.3: *The problem Eq. (2.25) has a unique solution for* $f \in V^* \otimes (S)^*$.

The example for the nonlinear SPDE Eq. (2.20) is deliberately kept simple. A generalisation to more general nonlinear SPDEs with strictly monotone, hemicontinuous, coercive operator is straightforward.

Chapter 3

Discretisation of Stochastic Partial Differential Equations

The first step in discretising a stochastic partial differential equation (SPDE) is to obtain a tractable representation in a countable number of mutually independent random variables; this is discussed in Section 3.1. Following this, the spatial semi-discretisation is presented (Section 3.2), which yields a system of equations with stochastic coefficients.

Two classes of techniques for the stochastic discretisation are considered: Methods based on a direct integration of statistics (Section 3.3) and methods based on series expansions in the stochastic dimensions (Section 3.4). The direct integration methods employ Monte Carlo integration and high-dimensional quadrature techniques. Series expansion discretisations are performed by orthogonal projections and by Galerkin methods in the stochastic dimensions, the latter of which are the focus of this thesis.

3.1 Representation in a Countable Number of Independent Random Variables

So far, all random properties have been expressed as functions on the abstract probability space $(\Omega, \mathcal{B}, P)$. A tractable representation is obtained by expressing all random fields in the SPDE in a countable number of mutually independent random variables. For details on such representations see the textbooks [62, 63, 174] or the reviews [80, 107, 156, 169].

Because of its favourable properties, the Karhunen–Loève expansion (KL-expansion) is used here for the random field discretisation (Section 3.1.1). The KL-expansion is computed by solving an eigenvalue problem and, in general, numerical techniques are required for this (see Section 3.1.2). Existing finite element

software will be used to discretise the eigenvalue problem (Section 3.1.3) and the error introduced by the FE discretisation will be discussed in Section 3.1.4. The resulting representation of the SPDE will be shown in Section 3.1.5.

3.1.1 The Karhunen–Loève expansion

The Karhunen–Loève expansion (KL-expansion) is discussed briefly. For more details see [1, 26, 57, 133, 174].

The KL-eigenvalue problem: Let $\kappa\colon R \times \Omega \longrightarrow \mathbb{R}$ be a random field with bounded covariance function $\operatorname{cov}_\kappa(x,y)$. Define the operator

$$\begin{array}{rcl} T\colon L^2(R) & \longrightarrow & L^2(R), \\ (Tu)(x) & := & \int_R \operatorname{cov}_\kappa(x,y)u(y)\,dy. \end{array} \tag{3.1}$$

As $\operatorname{cov}_\kappa$ is bounded and symmetric, T is a compact and selfadjoint Fredholm operator [177]. As the covariance function $\operatorname{cov}_\kappa$ is positive semi-definite, so is T. Hence, solutions of the eigenvalue problem

$$T\kappa_i = \lambda_i \kappa_i, \qquad \kappa_i \in L^2(R), i \in \mathbb{N}, \tag{3.2}$$

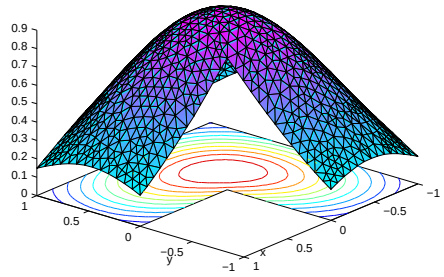
which is a Fredholm integral equation of the second kind, have the following properties [e.g. 177]: The eigenvalues λ_i are real and may be ordered as $\|T\| = \lambda_1 \geq \lambda_2 \geq \cdots \geq 0$. They satisfy $\|\operatorname{cov}_\kappa\|^2_{L^2(R\times R)} = \int_R \int_R |\operatorname{cov}_\kappa(x,y)|^2\,dx\,dy = \sum_i \lambda_i^2$ (T is a Hilbert-Schmidt operator). The eigenfunctions $\kappa_i(x)$ are mutually $L^2(R)$-orthogonal. If the covariance function is positive definite, then so is T and then the eigenfunctions are a basis of $L^2(R)$.

Usually, the KL-eigenmodes κ_i are oscillating functions, with finer and finer spatial fluctuations for growing i. This is demonstrated by Fig. 3.1: Plots (a–d) show eigenmodes of a random field on an L-shaped domain, where the covariance was chosen as in Eq. (2.11) with correlation length $\alpha = 1/4$. Plots (e,f) show two-dimensional slices of eigenmodes of a random field on a torus-shaped domain $R \subset R^3$, where the covariance function was chosen as $\operatorname{cov}_\kappa(x,y) = \exp(-\|x-y\|^2)$.

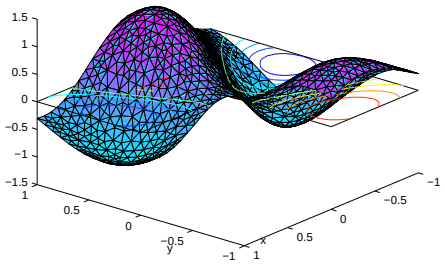
The Karhunen–Loève expansion is the series

$$\kappa(x,\omega) = \mu_\kappa(x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\,\kappa_i(x)\,\xi_i(\omega), \qquad \text{where} \tag{3.3}$$

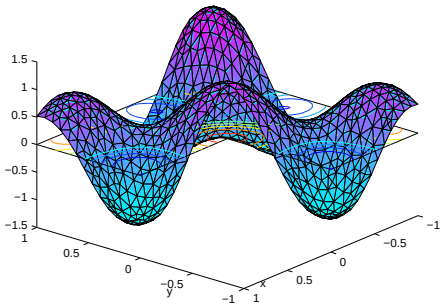
$$\xi_i(\omega) = \frac{1}{\sqrt{\lambda_i}}\big(\kappa(\cdot,\omega) - \mu_\kappa, \kappa_i\big)_{L^2(R)} = \frac{1}{\sqrt{\lambda_i}} \int_R \big(\kappa(x,\omega) - \mu_\kappa(x)\big)\kappa_i(x)\,dx. \tag{3.4}$$



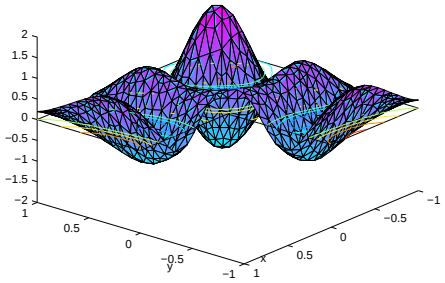
(a) First KL-mode, L-shaped domain



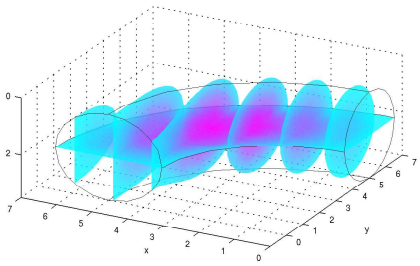
(b) 5-th KL-mode, L-shaped domain



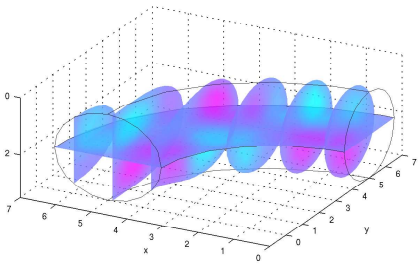
(c) 10-th KL-mode, L-shaped domain



(d) 20-th KL-mode, L-shaped domain



(e) First KL-mode, torus domain



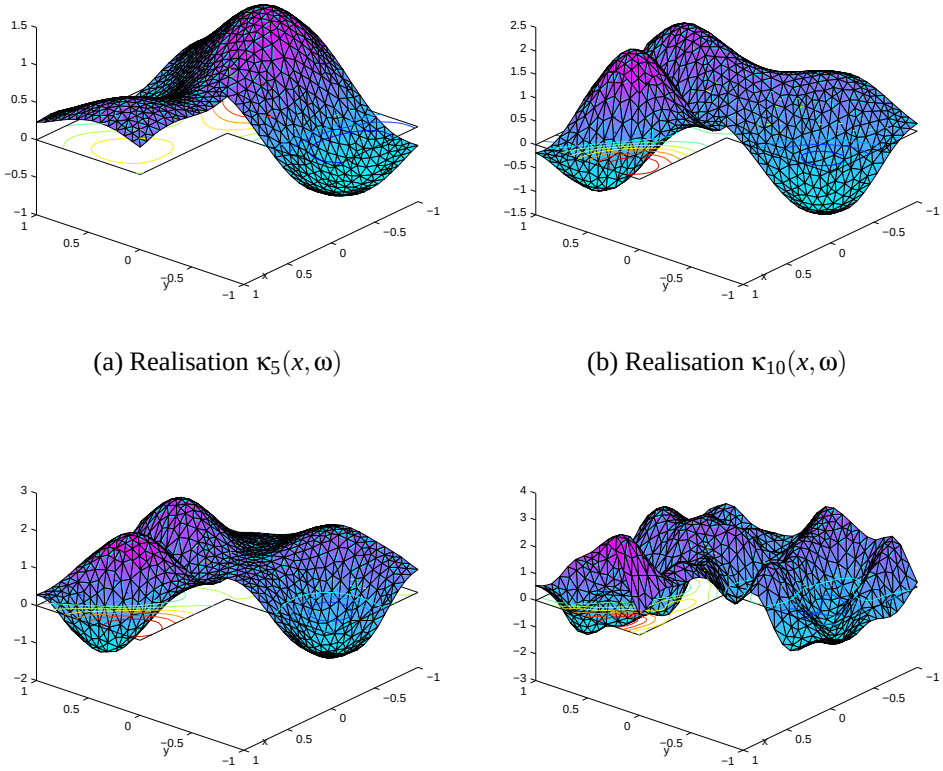
(f) 10-th KL-mode, torus domain

Figure 3.1: Examples of some Karhunen–Loève modes.

The random variables ξ_i are mutually uncorrelated and centred with unit variance [1, 57, 173, 174]. The truncated KL-expansion in m terms (see Fig. 3.2) is

$$\kappa_m(x,\omega) := \mu_\kappa(x) + \sum_{i=1}^{m} \sqrt{\lambda_i}\,\kappa_i(x)\,\xi_i(\omega). \tag{3.5}$$

If cov_κ is continuous, then $\mathrm{cov}_{\kappa_m}(x,y) = \sum_{i=1}^{m} \lambda_i \kappa_i(x)\kappa_i(y)$ converges absolutely and uniformly to $\mathrm{cov}_\kappa(x,y)$ on $R \times R$ by Mercer's Theorem [e.g. 177].



(a) Realisation $\kappa_5(x,\omega)$

(b) Realisation $\kappa_{10}(x,\omega)$

(c) Realisation $\kappa_{20}(x,\omega)$

(d) Realisation $\kappa_{100}(x,\omega)$

Figure 3.2: Realisations of $\kappa_m(x,\omega)$ for $m = 5, 10, 20, 100$.

Hence, κ_m converges to κ in variance uniformly, i.e. in $L^\infty(R) \otimes L^2(\Omega)$:

$$\sup_{x\in R} \mathbf{E}\left((\kappa(x) - \kappa_m(x))^2\right) = \sup_{x\in R} \sum_{i=m+1}^{\infty} \lambda_i \kappa_i(x)^2 \longrightarrow 0, \qquad \text{as } m \longrightarrow \infty. \quad (3.6)$$

The KL-expansion is an optimal linear approximation of κ in the sense that the error $\|\kappa - \mu_\kappa - \sum_{i=1}^m \sqrt{\lambda_i}\xi_i\kappa_i\|_{L^2(\Omega\times R)}$ is larger, if the functions $\sqrt{\lambda_i}\xi_i$ or the random variables κ_i are chosen differently [173]. However, nonlinear approximations may yield better approximations [30].

Finer and finer scales of spatial fluctuations are resolved by the approximation as more and more eigenfunctions are used in the truncated KL series. This is demonstrated in Fig. 3.2 by showing realisations of truncated KL-expansions of a Gaussian random field on an L-shaped region with covariance function Eq. (2.11) with correlation length $\alpha = 1/4$.

Our ultimate goal is to obtain a representation in mutually independent random variables. For Gaussian random fields $\kappa(x,\omega)$, the KL-expansion directly

yields such a representation: if $\kappa(x,\omega)$ is Gaussian, then the uncorrelated random variables $\xi_i(\omega)$ are Gaussian and hence mutually independent.

For non-Gaussian $\kappa(x,\omega)$, the random variables ξ_i are uncorrelated but not mutually independent. Section 3.1.5 discusses how to obtain a representation in independent random variables in this case.

Another complication arises for non-Gaussian $\kappa(x,\omega)$: the probability distributions of the ξ_i are then usually not analytically known. However, they may be obtained numerically from Eq. (3.4). This will be exploited in Section 4.1.1 to compute expansions of the non-Gaussian ξ_i in multivariate polynomials of independent Gaussian random variables (so-called polynomial chaos expansions).

3.1.2 The Discrete Karhunen–Loève expansion

Exact solutions of the Karhunen–Loève eigenvalue problem Eq. (3.2) are known only for special cases. Some exact solutions for $R \subset \mathbb{R}^1$ are given in [57, 173]. In general, the eigenvalue problem Eq. (3.2) needs to be solved numerically and standard techniques [e.g. 7, 66, 143, 165] may be used for this.

The Rayleigh-Ritz method may be used to discretise the Karhunen–Loève eigenvalue problem [e.g. 57, 148, 165] by projecting it onto a finite-dimensional subspace: Let $\mathcal{P}_{V^h}$ be the projection onto the n-dimensional space $V^h \subset L^2(R)$. By the Max-Min principle [e.g. 148], the largest eigenvalues of $\mathcal{P}^*_{V^h} T \mathcal{P}_{V^h}$ are lower bounds for the largest eigenvalues of T. Given a basis $N_1,\ldots,N_n \in V^h$ of V^h and defining $\boldsymbol{N}(x) = (N_1(x),\ldots,N_n(x))$, one may compute the eigenfunctions $\kappa_i^h(x) = \sum_{j=1}^n N_j(x)\kappa_{ji}^h = \boldsymbol{N}(x)\boldsymbol{\kappa}_i^h$ and the eigenvalues λ_i^h of the symmetric operator $\mathcal{P}^*_{V^h} T \mathcal{P}_{V^h}$ by the generalised matrix eigenproblem

$$\boldsymbol{W}\boldsymbol{\kappa}_i^h = \lambda_i^h \boldsymbol{M}\boldsymbol{\kappa}_i^h, \tag{3.7}$$

with

$$W_{ij} = \big(N_i, TN_j\big)_{L^2(R)} = \int_R \int_R N_i(x)\, \mathrm{cov}_\kappa(x,y)\, N_j(y)\, dx\, dy, \tag{3.8}$$

$$M_{ij} = \big(N_i, N_j\big)_{L^2(R)} = \int_R N_i(x)\, N_j(x)\, dx, \qquad i,j = 1,\ldots,n. \tag{3.9}$$

Standard iterative techniques [134, 153] may be employed to solve the generalised matrix eigenvalue problem Eq. (3.7). Here, it is computationally advantageous that the matrix $\boldsymbol{W}$ is symmetric positive semi-definite and that the Gram matrix $\boldsymbol{M}$ is symmetric positive definite.

For the numerical procedures discussed later, random fields will be represented by their truncated KL-expansion Eq. (3.5). For this, only the largest eigenvalues and the associated eigenfunctions need to be computed. Krylov subspace methods of Lanczos type may be employed for this.

An advantage of the Krylov subspace methods is that they do not require the matrices $\boldsymbol{W}$ and $\boldsymbol{M}$ in assembled form. This is especially advantageous as the matrix $\boldsymbol{W}$ is a dense matrix (the matrix $\boldsymbol{M}$ is usually sparse). Only routines for computing the matrix-vector products with $\boldsymbol{W}$ and with $\boldsymbol{M}$ and a facility for solving linear equations involving the matrix $\boldsymbol{M}$ are needed. Efficient implementations of Krylov subspace methods are readily available: the numerical examples of this thesis were computed using the software library ARPACK [95, 104].

3.1.3 Finite Element Discretisation of the Karhunen–Loève Expansion

In the further presentation, the space V^h will be chosen as a space of finite element shape functions on the region $R \subset \mathbb{R}^d$. A discussion of finite element procedures for eigenvalue problems may be found in [e.g. 165].

Throughout the thesis, it will be assumed that some finite element (FE) software is available that will be called the *deterministic code*. This software may be used in a black-box fashion to obtain the matrices $\boldsymbol{W}$ and $\boldsymbol{M}$.

Writing the FE shape functions as a vector $\boldsymbol{N}(x) = (N_1(x), \ldots, N_n(x))$, with $N_i \colon R \to \mathbb{R}$, one may approximate the covariance function as

$$\operatorname{cov}_\kappa(x,y) \approx \operatorname{cov}_\kappa^h(x,y) := \sum_{i=1}^{n} \sum_{j=1}^{n} N_i(x) C_{ij} N_j(y) = \boldsymbol{N}(x) \boldsymbol{C} \boldsymbol{N}(y)^T, \tag{3.10}$$

where the matrix $\boldsymbol{C}$ is chosen such that $\operatorname{cov}_\kappa^h$ interpolates $\operatorname{cov}_\kappa$.

The Gram matrix $\boldsymbol{M}$ is a special case of the "mass matrix", which most FE-codes are able to compute, and $\boldsymbol{W}$ may be approximated as

$$\begin{aligned} \boldsymbol{W} &\approx \int_R \int_R \boldsymbol{N}(x)^T \operatorname{cov}_{\kappa^h}(x,y) \boldsymbol{N}(y) \, dx \, dy && (3.11) \\ &= \int_R \int_R \boldsymbol{N}^T(x) \boldsymbol{N}(x) \boldsymbol{C} \boldsymbol{N}(y)^T \boldsymbol{N}(y) \, dx \, dy && (3.12) \\ &= \boldsymbol{M} \boldsymbol{C} \boldsymbol{M}. && (3.13) \end{aligned}$$

Instead of the discrete eigenproblem Eq. (3.7), the eigenvalue problem

$$\boldsymbol{M} \boldsymbol{C} \boldsymbol{M} \boldsymbol{\kappa}_i^h = \lambda_i^h \boldsymbol{M} \boldsymbol{\kappa}_i^h. \tag{3.14}$$

is solved; the error introduced by this is discussed in the next section.

As $\boldsymbol{M}$ is regular, Eq. (3.14) may be rewritten as the unsymmetric standard eigenvalue problem $\boldsymbol{C} \boldsymbol{M} \boldsymbol{\kappa}_i = \lambda_i \boldsymbol{\kappa}_i$. It may be computationally advantageous [153] to rewrite this as a symmetric eigenvalue problem: Using the Cholesky decomposition of the symmetric positive definite matrix $\boldsymbol{M} = \boldsymbol{L} \boldsymbol{L}^T$, one may rewrite

Eq. (3.14) as the symmetric eigenvalue problem $\boldsymbol{L}^T \boldsymbol{C} \boldsymbol{L} \boldsymbol{y}_i = \lambda_i \boldsymbol{y}_i$ with $\boldsymbol{\kappa_i} = \boldsymbol{L}^{-T} \boldsymbol{y}_i$; see [153].

An existing finite element code may thus be used in a black-box fashion to discretise the KL-eigenvalue problem if it provides a routine for the matrix-vector product with the mass matrix. If a routine for factoring the mass-matrix is available, then the eigenvalue problem Eq. (3.14) may be solved as a symmetric standard eigenvalue problem. Otherwise, an unsymmetric standard eigenvalue problem needs to be computed.

Solving Eq. (3.14), e.g. by the Krylov subspace methods discussed in the previous section, yields an approximation of the truncated KL-series

$$\kappa_m(x,\omega) \approx \kappa_m^h(x,\omega) = \mu_\kappa^h(x) + \sum_{i=1}^{m} \sqrt{\lambda_i^h} \boldsymbol{N}(x) \boldsymbol{\kappa}_i^h \xi_i(\omega), \tag{3.15}$$

where $\mu_\kappa^h(x)$ is the interpolant of the mean function $\mu_\kappa(x)$ in the FE basis.

Example 3.1: The KL-eigenmodes shown in Fig. 3.1 (a–d) and the realisations of the truncated KL-expansions shown in Fig. 3.2 (a–d) were computed by a finite element discretisation of an L-shaped domain by triangular elements using piecewise linear shape functions with approximately 1.000 degrees of freedom.

The KL-eigenmodes shown in Fig. 3.1 (e–f) were computed by a finite element discretisation of a torus-shaped domain $R \subset \mathbb{R}^3$ by tetrahedron elements using piecewise linear shape functions with approximately 4.000 degrees of freedom.

3.1.4 The Error Introduced by the Finite Element Discretisation of the Karhunen–Loève Expansion

Let us now analyse the error introduced in the previous section. For simplicity, assume that the $N_1, \ldots, N_n$ are a nodal basis [127, 165] in the nodes $x_1, \ldots, x_n \in R$. Particularly, assume that the ansatz functions $N_i \in L^2(R)$ are a partition of unity with $N_i(x) \geq 0, i = 1, \ldots, n$, and that

$$N_i(x_j) = \delta_{ij}, \qquad i,j = 1, \ldots, n. \tag{3.16}$$

The interpolant of $\kappa(x,\omega)$ in the FE basis is then

$$\kappa^h(x,\omega) = \sum_{i=1}^{n} N_i(x) \zeta_i(\omega) = \boldsymbol{N}(x) \boldsymbol{\zeta}(\omega), \tag{3.17}$$

where $\zeta_i(\omega) := \kappa(x_i,\omega),\ i = 1, \ldots, n$. The superscript $h > 0$ denotes the largest finite element diameter, i.e. $h = \max_i \operatorname{diam}(\operatorname{supp} N_i)$. Apparently, the covariance function of κ^h is

$$\operatorname{cov}_{\kappa^h}(x,y) = \boldsymbol{N}(x) \boldsymbol{C} \boldsymbol{N}(y)^T, \tag{3.18}$$

with $C_{ij} = \mathrm{cov}_\kappa(x_i, x_j)$, $i, j = 1, \ldots, n$. As the FE shape functions are now assumed to be a nodal basis, the interpolant of the covariance function defined in the previous section is $\mathrm{cov}^h_\kappa(x,y) = \mathrm{cov}_{\kappa^h}(x,y)$. Hence, computing the FE discretisation of the KL-expansion as in the previous section is equivalent to computing the exact KL-expansion of κ^h if the N_i are a nodal basis. The error introduced by the approximation Eq. (3.11) for $\boldsymbol{W}$ may hence be analysed by investigating how well κ^h approximates κ.

Standard results [127] hold for the approximation of the mean $\mu_\kappa(x)$ in the finite element ansatz. To concentrate on the error of the stochastic part, assume that $\mu_\kappa(x) = \mu_{\kappa^h}(x) = 0$ for $x \in R$ and thus $\mathrm{cov}_\kappa(x,y) = \mathbf{E}\left(\kappa(x,\omega)\kappa(y,\omega)\right)$.

The approximation error in variance is

$$\begin{aligned}
&\mathrm{var}(\kappa(x,\omega) - \kappa^h(x,\omega)) \\
&= \mathbf{E}\left(\kappa(x)^2\right) - 2\sum_{i=1}^{n} \mathbf{E}\left(\kappa(x)\kappa(x_i)\right) N_i(x) + \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{E}\left(\kappa(x_i)\kappa(x_j)\right) N_i(x) N_j(x) \\
&= e_1(x) + e_2(x),
\end{aligned}$$

where, because of $\sum_{i=1}^n N_i(x) = 1$,

$$e_1(x) = \sum_{i=1}^{n} N_i(x)\left(\mathrm{cov}_\kappa(x,x) - \mathrm{cov}_\kappa(x,x_i)\right), \tag{3.19}$$

$$e_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} N_j(x) N_i(x)\left(\mathrm{cov}_\kappa(x_i,x_j) - \mathrm{cov}_\kappa(x_i,x)\right). \tag{3.20}$$

Observe that $e_1(x) \le \sum_{i=1}^n N_i(x) \sup_{|x-y|<h} |\mathrm{cov}_\kappa(x,x) - \mathrm{cov}_\kappa(x,y)|$, as $N_i(x) = 0$ for $|x - x_i| > h$. The N_i are a partition of unity and hence $e_1(x) \le \Delta(h)$, where

$$\Delta(h) := \sup_{\substack{y_1, y_2 \in R \\ |y_1 - y_2| \le h}} |\mathrm{cov}_\kappa(y_1,y_1) - \mathrm{cov}_\kappa(y_1,y_2)| \tag{3.21}$$

measures the rate of decay of the covariance function. Similarly, one may see that $e_2(x) \le \Delta(h)$. Therefore, the approximation converges uniformly in variance,

$$\sup_{x \in R} \mathrm{var}(\kappa(x,\omega) - \kappa^h(x,\omega)) \le 2\Delta(h). \tag{3.22}$$

To illustrate this estimate, assume that κ is homogeneous and isotropic and hence $C(x,y) = c(\|x-y\|)$ with an appropriate function $c\colon \mathbb{R} \to \mathbb{R}$. Then $\Delta(h) = c(0) - c(h)$ and for a given h, the approximation is the better the slower the covariance function decays. This estimate may be employed to choose the appropriate mesh size for the discretisation of a random field, e.g. by selecting h such that $\Delta(h)$ is smaller than some threshold.

The KL-expansion of κ is approximated here by computing the exact KL-expansion of $\kappa^h(x,\omega)$. It is sometimes noted [e.g. 72, 169] that doing so reduces the KL-expansion technique to the interpolation method [e.g. 80, 108, 169], thus losing the optimality of the KL-expansion. But as shown here, the numerical solution converges to the analytical solution for finer and finer spatial discretisations. Analogous to the standard theory of finite element interpolations, exploiting the smoothness of the covariance function might lead to more refined estimates.

The KL-expansion yields an optimal linear approximation and the quality of the discretised KL-expansion increases for a decreasing mesh size h. It may thus be advantageous to perform the random field discretisation and the spatial discretisation of the stochastic partial differential equation (see Section 3.2) on different meshes. The mesh for the KL-discretisation should be chosen as fine as required for the random field and after solving the KL-eigenproblem the result may be interpolated to the mesh for the spatial discretisation of the SPDE.

It is sometimes stated that the costs in numerically solving the eigenvalue-problem are a disadvantage of the KL-expansion. But in the author's experience, these costs are not the limiting factor in stochastic finite element simulations. According to Section 3.1.1, the KL-expansion minimises (for a Gaussian random field) the number of independent random variables required for an approximation with a given error. Each independent random variable will be seen to introduce one dimension into the discretised problem and the minimisation of the dimensions outweighs the costs of solving the KL-eigenproblem by far.

Only the FE discretisation error was analysed here. In practice, the number of FE shape functions is larger than the number of terms kept in the truncated KL-expansion. The resulting truncation error is usually larger than the FE discretisation error, but this is not investigated here. If only m terms of the KL-expansion are kept, then $\operatorname{var}(\kappa^h(x)-\kappa^h_m) = \sum_{i=m+1}^{n} \lambda_i^h (\boldsymbol{N}(x)\boldsymbol{\kappa}_i^h)^2$. This truncation error may be evaluated by computing all eigenvalues of Eq. (3.14) and together with Eq. (3.22) this would give an à posteriori bound for the total truncation error. Also note that the best obtainable truncation error for any m-term discretisation of the KL-expansion is $\sum_{i=m+1}^{\infty} \lambda^2 = \|\operatorname{cov}_\kappa\|_2^2 - \sum_{i=0}^{m} \lambda_i^2$ because $\|\operatorname{cov}_\kappa\|_2^2 = \sum_{i=0}^{\infty} \lambda_i^2$.

To obtain à priori estimates for the truncation error, the decay of the eigenvalues λ_i for growing i needs to be analysed depending on the domain R and on the covariance function $\operatorname{cov}_\kappa$, but the author is not aware of theoretical results on this. Numerical tests [72] and analogies with the Fourier transform indicate that the number of KL-terms to be kept for a given error is the smaller the smoother the covariance function is and the slower the covariance function $\operatorname{cov}_\kappa(x,y)$ decays with growing $\|x-y\|$, as then λ_n decays quickly as $n \to \infty$.

3.1.5 Representation in Independent Random Variables

The stochastic partial differential equation will now be represented in a countable number of mutually independent random variables $\theta_1, \theta_2, \theta_3, \ldots \in L^2(\Omega)$. The sequence of these mutually independent random variables will be called $\theta = (\theta_1, \theta_2, \ldots)$.

In general, it is difficult to compute a representation of non-Gaussian random fields $\kappa(x, \omega)$ in independent random variables. A standard technique is the Rosenblatt transform [116, 151]. It might be used to represent the uncorrelated non-Gaussian random variables $\xi_1, \xi_2, \ldots$ as transformations $\xi_i = \xi_i(\theta_1, \ldots, \theta_i)$, $i = 1, \ldots, m$ of independent Gaussian random variables $\theta_1, \theta_2, \ldots$. While this is a general technique, it requires a representation of the joint probability distributions of the $\xi_1, \xi_2, \ldots$ as products of conditional probability distributions.

A representation in mutually independent random variables may be easily obtained for random fields modelled as in Section 2.2:

- Let $\gamma(x, \omega)$ be a Gaussian random field and denote by ν_i and $\gamma_i(x), i = 1, 2, \ldots$, its KL-eigenvalues and KL-eigenmodes. Then its KL-expansion is $\gamma(x, \omega) = \mu_\gamma(x) + \sum_{i=1}^{\infty} \sqrt{\nu_i}\gamma_i(x)\theta_i(\omega)$ with mutually independent standard Gaussian random variables $\theta = (\theta_1, \theta_2, \ldots)$. The dependence on ω is omitted and the random field is written as $\gamma(x, \theta) = \mu_\gamma(x) + \sum_{i=1}^{\infty} \sqrt{\nu_i}\gamma_i(x)\theta_i$.

- Let κ be a transformation $\kappa(x, \omega) = \phi(x, \gamma(x, \omega))$ of a Gaussian random field $\gamma(x, \omega)$ with KL-expansion as above. It may be represented in the independent random variables as $\kappa(x, \theta) = \phi(x, \mu_\gamma(x) + \sum_{i=1}^{\infty} \sqrt{\nu_i}\gamma_i(x)\theta_i)$, where again the dependence on ω is omitted. This representation will be used in Section 4.1.1.2 to expand the uncorrelated κ_i in multivariate polynomials of Gaussian random variables (the so-called polynomial chaos).

- For a random field defined by Eq. (2.7) a representation in mutually independent random variables is given by the definition.

It will be assumed from now on that all random fields are represented as functions $\kappa(x, \omega) = \kappa(x, \theta(\omega))$ of mutually independent random variables $\theta = (\theta_1, \theta_2, \ldots) \in (L^2(\Omega))^{\mathbb{N}}$. For the random field models of Section 2.2 it has been shown here how such representations may be obtained. For other types of random fields, other techniques must be used, e.g. the Rosenblatt transform mentioned above.

For convenience, the dependence on ω will be omitted and $\kappa(x, \theta)$ will be written for a random field. If its variance var_κ is finite, then $\kappa(x, \theta)$ is a random field on the probability space $L^2(\Omega, \Sigma(\theta), P)$, where $\Sigma(\theta)$ denotes the σ-algebra generated by $\theta = (\theta_1, \theta_2, \ldots)$. By representing all random fields in the SPDE as functions of the sequence of mutually independent random variables $\theta = (\theta_1, \theta_2, \ldots)$,

the probability space of the SPDE formulation may be replaced by the probability space $(\Omega, \Sigma(\theta), P)$ and the space (S) of admissible stochastic functions is $(S) = L^2(\Omega, \Sigma(\theta), P)$ (recall that (S) was assumed to be separable). The solution is then also a random field on this probability space and may be written as a function $u = u(x, \theta)$ of the mutually independent random variables $\theta = (\theta_1, \theta_2, \ldots)$.

The representation in a countable number of independent random variables developed here will be employed by some of the stochastic discretisation techniques (see sections 3.3 and 3.4.2) to approximate the SPDE in a finite number of independent random variables $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$.

3.2 Spatial Discretisation

Almost any technique may be used for the spatial discretisation, for example finite differences or finite elements. It is assumed that some existing simulation software, named the *deterministic code* or the *deterministic solver*, performs the spatial discretisation or the solution of the deterministic analogon of the SPDE.

For simplicity, a spatial discretisation by finite elements is assumed; the generalisation to other discretisation techniques is straightforward. For introductions to the finite element method (FEM) see e.g. [27, 127, 165, 190].

The spatial domain $R \subset \mathbb{R}^d$ is discretised by finite elements with shape functions $N_1(x), \ldots, N_n(x) \in V^h \subset V$, where V^h is a finite element ansatz space with maximum element diameter $h > 0$, which is contained in the space V of admissible spatial functions. For example, the space of admissible spatial functions is $V = \mathring{H}^1(R)$ for the second-order linear elliptic SPDE Eq. (2.13) and $V = \mathring{W}_4^1(R)$ for the second-order nonlinear elliptic SPDE Eq. (2.21). Note that the FE ansatz space V^h may be different from the finite element space used in Section 3.1.3 for the random field discretisation.

Collecting the shape functions in the vector $\boldsymbol{N}(x) = (N_1(x), \ldots, N_n(x))$ and the stochastic degrees of freedom in the vector $\boldsymbol{u}(\theta) = (u_1(\theta), \ldots, u_n(\theta))^T$ with $\theta = (\theta_1, \theta_2, \ldots)$, one may expand the solution as

$$u(x, \theta) \approx u^h(x, \theta) := \sum_{i=1}^{n} u_i(\theta) N_i(x) = \boldsymbol{N}(x) \boldsymbol{u}(\theta). \tag{3.23}$$

This is similar to the method of lines for instationary boundary value problems, where the degrees of freedom would be time-dependent.

Inserting Eq. (3.23) into the SPDE and applying Galerkin conditions in the space $V^h \otimes (S)$ yields the semi-discretisation of the SPDE, which requires to

$$\begin{aligned} &\text{find } u^h \in V^h \otimes (S) \text{ such that} \\ &a(u^h, v^h) = f(v^h) \qquad \text{for all } v^h \in V^h \otimes (S). \end{aligned} \tag{3.24}$$

This is the weak formulation of the system of equations

$$\boldsymbol{r}(\boldsymbol{u}(\theta),\theta) = 0, \tag{3.25}$$

where the residual $\boldsymbol{r}(\boldsymbol{u}(\theta),\theta)$ is a function with random coefficients. If the SPDE is linear, then the residual may be written as

$$\boldsymbol{r}(\boldsymbol{u}(\theta),\theta) := \boldsymbol{K}(\theta)\boldsymbol{u}(\theta) - \boldsymbol{f}(\theta) = 0, \tag{3.26}$$

with a random matrix $\boldsymbol{K}$ and a random vector $\boldsymbol{f}$.

Example 3.2: As an example, the spatial discretisation of the linear elliptic SPDE Eq. (2.13) is

$$\boldsymbol{K}(\theta) = \int_R \nabla \boldsymbol{N}(x)\kappa(x,\theta)(\nabla \boldsymbol{N}(x))^T\, dx, \tag{3.27}$$

$$\boldsymbol{f}(\theta) = \int_R f(x,\theta)(\boldsymbol{N}(x))^T\, dx. \tag{3.28}$$

The residual of the nonlinear elliptic SPDE, Eq. (2.20), is

$$\boldsymbol{r}(\boldsymbol{u}(\theta),\theta) = \int_R \nabla \boldsymbol{N}(x) G(\boldsymbol{N}(x)\boldsymbol{u}(\theta))\, dx - \boldsymbol{f}(\theta),$$

where G is the Nemicky operator defined in Eq. (2.23) and where $\boldsymbol{f}(\theta)$ is given by Eq. (3.28).

3.3 Direct Integration of Statistics

From the solution of an SPDE one usually wants to compute statistics. Numerical integration techniques may be used for this if the SPDE is approximated in a finite number of independent random variables. This approximation and the stability of the resulting perturbation are discussed in Section 3.3.1.

Statistics and their direct integration are discussed in Section 3.3.2. The resulting integrals are high-dimensional and appropriate numerical integration methods are discussed in Section 3.3.3 and in more detail in the Appendix B. The techniques for high-dimensional integration presented there will also be used as workhorses for other stochastic discretisation techniques.

3.3.1 Approximation in a Finite Number of Independent Random Variables

The direct computation of statistics, e.g. by Monte Carlo methods, requires to approximate the SPDE in a finite number of independent random variables. This approximation is performed here by keeping only a finite subset $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ of the mutually independent random variables obtained in Section 3.1.5.

The approximation of a random field $\kappa(x,\theta)$ in these m random variables will be called $\kappa(x,\boldsymbol{\theta})$. Random fields modelled as a transformation $\kappa(x,\theta) = \phi(x,\gamma(x,\theta))$ of a Gaussian random field γ will be approximated either by their KL-approximation or as $\kappa(x,\boldsymbol{\theta}) := \phi(x,\gamma(x,\boldsymbol{\theta}))$, where $\gamma(x,\boldsymbol{\theta})$ is the truncated KL-expansion of γ. Other approximation-techniques for non-Gaussian random fields may also be used, but these will not be considered here; see [e.g. 62].

A random field in m independent random variables may be identified [8, 12, 34] with a random field on the probability space $(\Omega^{(m)}, \mathcal{B}^{(m)}, P_m)$, where $\Omega^{(m)} = \Omega_1 \times \cdots \times \Omega_m \subset \mathbb{R}^m$ with $\Omega_i := \text{range}(\theta_i) = \theta_i(\Omega)$, where $\mathcal{B}^{(m)}$ is the Borel σ-algebra of $\Omega^{(m)}$, and where P_m is the probability distribution of the random vector $\boldsymbol{\theta}$ (the measure induced on its range). One may identify $L^2(\Omega, \Sigma(\boldsymbol{\theta}), P) \cong L^2(\Omega^{(m)}, \mathcal{B}^{(m)}, P_m)$ and regard the approximated random fields as maps from $R \times \Omega^{(m)} \subset \mathbb{R}^{d+m}$ to the real numbers. Each of the independent random variables $\theta_1, \ldots, \theta_m$ may thus be seen as an axis of a coordinate system of this $d+m$-dimensional space [8, 12, 34]. Elementary events from this finite-dimensional sample space will be written as $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m) \in \Omega^{(m)}$, where $\omega_i \in \Omega_i$, $i = 1, \ldots, m$.

Replacing all random fields in the SPDE by their approximations in the m independent random variables $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ yields an approximation of the SPDE on the probability space $(\Omega, \Sigma(\boldsymbol{\theta}), P)$. This perturbed SPDE has as solution a random field $u(x,\boldsymbol{\theta})$ from the space $V \otimes L^2(\Omega, \Sigma(\boldsymbol{\theta}), P)$ [8, 12, 34], where V is the

space of admissible spatial functions (see Section 2.3.1). Note that this perturbed SPDE may be identified with a PDE on the domain $R \times \Omega^{(m)} \subset \mathbb{R}^{d+m}$ and that the solution may be identified with a function $u\colon R \times \Omega^{(m)} \to \mathbb{R}$ [8, 12, 34].

The stability of this approximation is discussed here exemplary for the linear SPDE Eq. (2.13) with associated bilinear form $a_\kappa(u,v)$ defined in Eq. (2.17).

Stability with respect to perturbations of the right hand side f in $L^2(R) \otimes L^2(\Omega)$ was shown in Theorem 2.2. The approximation process is stable if the truncated KL-expansion of f is used as the KL-expansion converges in $L^2(R) \otimes L^2(\Omega)$ (see Section 3.1.1),

According to Theorem 2.2, the solution depends continuously on perturbations of κ in $L^\infty(R) \otimes L^2(\Omega)$ if the perturbed bilinear form is coercive and bounded. If the covariance function cov_κ is continuous, then the KL-expansion of κ converges in $L^\infty(R) \otimes L^2(\Omega)$ (see Section 3.1.1). Hence, approximating κ by its truncated KL-expansion is stable if additionally the truncated KL-expansion is uniformly bounded away from zero and from above.

Care must be taken if the material κ is approximated by its truncated KL-expansion: Babuška and Chatzipantelides [8] analyse the case where κ is a random field with covariance function $\mathrm{cov}_\kappa(x,y) = \exp(-c\|x-y\|)$ and where $a_\kappa(u,v)$ is coercive. Approximating κ by its m-term truncated KL-expansion κ_m, they show that the bilinear form $a_{\kappa_m}(u,v)$ is not coercive if m is sufficiently large and that the perturbed SPDE is unsolvable. Thus, one should check numerically whether the coercivity conditions are violated after computing the truncated KL-expansion.

The approximation of the SPDE may be guaranteed to be stable if the random field model of Section 2.2.4.1 is used. Let κ be defined as a transformation $\kappa(x,\theta) = \phi(x,\gamma(x,\theta))$ of a Gaussian random field γ. If κ is approximated as $\kappa_m(x,\boldsymbol{\theta}) := \phi(x,\gamma_m(x,\boldsymbol{\theta}))$, where γ_m is the truncated KL-expansion of γ, then the perturbed bilinear form may be guaranteed to be coercive and bounded by an appropriate choice of the transformation ϕ. Additionally, one may guarantee that κ_m converges to κ in $L^\infty(R) \otimes L^2(\Omega)$; the following Lemma shows pointwise convergence:

Lemma 3.1: *Let $\gamma(x,\theta)$ be a centred Gaussian random field on a region $R \subset \mathbb{R}^d$ with continuous covariance function cov_γ and with unit variance. Let γ_m be its m-term truncated KL-expansion. Let $F_{\kappa(x)}, x \in R$, be continuous probability distribution functions, let $\kappa(x,\theta) := \phi(x,\gamma(x,\theta)) := F_x^{-1} \circ \mathrm{erf}\,\gamma(x,\theta)$, and let*

$$\kappa_m(x,\theta) := \phi(x,\gamma_m(x,\theta)).$$

Let there be an $\varepsilon > 0$ such that $\mathbf{E}\left(\kappa(x,\theta)^{2+\varepsilon}\right) < \infty, \mathbf{E}\left(\kappa_m(x,\theta)^{2+\varepsilon}\right) < \infty$, and $\mathbf{E}\left(|\kappa(x,\theta)\kappa_m(x,\theta)|^{1+\varepsilon}\right) < \infty$ for all $x \in R$ and for all $m > 0$. Then for each $x \in R$

$$\|\kappa(x,\theta) - \kappa_m(x,\theta)\|_{L^2(\Omega)} \to 0, \quad m \to \infty.$$

PROOF: The mean $\mu_\kappa(x,\theta)$ may be chosen by adding a deterministic function to ϕ, hence one may assume without loss of generality that $\mu_\kappa(x) = 0$ and that $\mu_{\kappa_m}(x) = 0$ for all m.

The pointwise convergence is shown in [24, Lemma 3], according to which the covariance of two random variables defined as transformation of two Gaussian random variables depends continuosly on the covariance of these two Gaussian random variables. According to [24, Lemma 3], $\mathbf{E}\left(\kappa_m(x,\theta)^2\right) \to \mathbf{E}\left(\kappa^2(x,\theta)\right)$ and $\mathbf{E}\left(\kappa(x,\theta)\kappa_m(x,\theta)\right) \to \mathbf{E}\left(\kappa^2(x,\theta)\right)$ for $m \to \infty$. Therefore, $\|\kappa(x,\theta) - \kappa_m(x,\theta)\|^2 = \mathbf{E}\left(\kappa(x,\theta)^2\right) - 2\,\mathbf{E}\left(\kappa\kappa_m(x,\theta)\right) + \mathbf{E}\left(\kappa_m(x,\theta)\right) \to 0$ for $m \to \infty$. □

The previous Lemma shows pointwise convergence. To obtain convergence in $L^\infty(R) \otimes L^2(\Omega)$ requires that the convergence is independent of x. As the convergence speed depends on the particular choice of the transformation ϕ, this will not be investigated here in detail. In the experiments used here, the material is defined as $\kappa(x,\gamma(x,\theta)) = \phi(x,\gamma) = \mu(x) + \phi'(\gamma(x,\theta))$, where $\mu(x)$ denotes the mean of $\kappa(x,\theta)$ and where $\phi'(\gamma(x,\theta))$ is a centred random variable with a probability distribution that is independent of $x \in R$. Then κ_m converges to κ in $L^\infty(R) \otimes L^2(\Omega)$. Hence, replacing κ by κ_m yields a stable approximation of the SPDE in a finite number of independent random variables.

3.3.2 Direct Integration of Statistics

The representation in a finite number of independent random variables is now used to compute statistics of the solution. For simplicity, only univariate statistics are considered; the generalisation to multivariate statistics is straightforward.

Properties of interest may be, for example, the mean of the response $\mu_u(x) = \mathbf{E}\left(u(x,\theta)\right)$, its variance $\mathrm{var}_u(x) = \mathbf{E}\left((u(x,\theta) - \mu_u(x))^2\right)$, or its probability to exceed some threshold, $p_u(x) = \mathrm{prob}\{u(x,\theta) > u_0\} = \mathbf{E}\left(\chi_{(u_0,\infty)}(u(x,\theta))\right)$. With appropriate functionals $s(x,\cdot)$, these univariate statistics may be written as integrals with respect to the probability measure P as

$$s_u(x) := \mathbf{E}\big(s(x,u(x,\theta))\big) = \int_\Omega s(x,u(x,\theta(\omega)))\,dP(\omega). \tag{3.29}$$

As shown in Section 3.3.1, the SPDE may be approximated in a finite number of mutually independent random variables $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_m)$. Thus, one may compute $s_u(x) \approx s_u^m(x) := \mathbf{E}\big(s(x,u(x,\boldsymbol{\theta}))\big) = \int_{\Omega^{(m)}} s(x,u(x,\boldsymbol{\omega}))\,dP_{\boldsymbol{\theta}}(\boldsymbol{\omega})$.

The θ_i are independent and hence, by Fubini's lemma,

$$s_u^m(x) = \int_{\Omega_1} \cdots \int_{\Omega_m} \underbrace{s(x,\boldsymbol{\omega},\boldsymbol{N}(x)\boldsymbol{u}(\boldsymbol{\omega}))}_{=:\psi(\boldsymbol{\omega})}\,dP_{\theta_1}(\omega_1)\cdots dP_{\theta_m}(\omega_m), \tag{3.30}$$

where $\Omega_i := \theta_i(\Omega)$ is the range of θ_i, $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m) \in \Omega^{(m)}$, and $dP_{\theta_i}(\omega_i)$ is the probability distribution of θ_i (see Section 3.3.1).

For simplicity, the integrand is called ψ and the integral is written as

$$s = \int_{\Omega} \psi(\boldsymbol{\omega})\, dP_m(\boldsymbol{\omega}) = \int_{\Omega_1} \cdots \int_{\Omega_m} \psi(\boldsymbol{\omega})\, dP_{\theta_1}(\omega_1) \cdots dP_{\theta_m}(\omega_m). \tag{3.31}$$

Numerical techniques for evaluating this high-dimensional integral are discussed in Section 3.3.3 and in more detail in the Appendix B. The numerical integration techniques compute an approximation s_Z of the integral by evaluating the integrand in Z integration points $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(Z)} \in \Omega^{(m)}$ and by linearly combining the results with weights $w_1, \ldots, w_Z \in \mathbb{R}$ as

$$s \approx s_Z := \sum_{z=1}^{Z} w_z \psi(\boldsymbol{\omega}^{(z)}). \tag{3.32}$$

The evaluation of Eq. (3.32) may be performed as follows:

(1) Select the integration points $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(Z)} \in \Omega^{(m)}$ and weights $w_1, \ldots, w_Z$ according to the integration rule. As example, for a Monte Carlo integration the $\boldsymbol{\omega}^{(i)}$ are chosen as randomly distributed according to the probability distribution of the $\boldsymbol{\theta}$ and all weights are chosen as $w_i = 1/Z$.
(2) Each integration point $\boldsymbol{\omega}^{(z)}$ yields a realisation of the SPDE. After solving it, the integrand $\psi(\boldsymbol{\omega}^{(z)})$ can be evaluated.

Note that these computations may be sped up by a simple technique: If the distance in $\Omega^{(m)}$ between two integration points is small, then the corresponding realisations of the SPDE are similar. For non-linear SPDEs or for linear SPDEs solved by iterative methods, it may be observed in numerical experiments that the computation times can be reduced by taking the solution at an integration point as initial guess when solving realisations of the SPDE at close-by integration points.

3.3.3 High-Dimensional Integration

What integration rule one should use for the evaluation of the high-dimensional integral Eq. (3.42) depends on the dimensions m and on the integrand. This section briefly sketches Monte Carlo methods [e.g. 22], quasi-Monte Carlo methods [e.g. 22, 121], full tensor quadrature and expands somewhat on sparse (Smolyak) tensor quadrature [163]; for more details see Appendix B and Caflish's review [22] and the references therein.

Monte Carlo methods (MC-methods) choose the integration points as Z independent realisations $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_Z \in \Omega^{(m)}$ of the random variables $\boldsymbol{\theta}$ and use constant weights $w_i = Z^{-1}$.

As the integration points are chosen randomly, MC-methods are probabilistic: both s_Z and the error $s - s_Z$ are random variables. For large Z, the error is approximately $\sigma_\psi Z^{-1/2}\mathcal{N}(0,1)$, where $\mathcal{N}(0,1)$ is a standard-distributed Gaussian random variable, and where σ_ψ is the standard deviation of the integrand. As the error is of order $O(\sigma_\psi Z^{-1/2})$, MC-methods converge slowly: for instance, the error is reduced by one order of magnitude if the number of evaluations is increased by two orders.

MC-methods are well suited for integrands with small variance and low accuracy requirements. In applications, their efficiency is usually increased somewhat by variance reduction and importance sampling; see e.g. [22, 156, 157] and the references therein. The significant advantage of MC-methods is their dimension independent convergence rate; the efficiency of the other integration techniques decreases with increasing dimensions.

An alternative are quasi-Monte Carlo methods (QMC-methods); e.g. see the review by Caflish [22] and Niederreiter's monograph [121]. Informally speaking, quasi-Monte Carlo methods choose the sequence of integration points such that "for any number of points the integral $\mathbf{E}(1)$ is approximated well by the sequence". Such sequences are called quasi-random numbers or low discrepancy sequences [121]; for how to generate quasi-random numbers for arbitrary random vectors $\boldsymbol{\theta}$ see e.g. [69] and the references therein. The usual QMC-methods have an error of $O\big(\|\psi\|_{BV} Z^{-1}(\log Z)^m\big)$, where $\|\psi\|_{BV}$ denotes the bounded variation norm. If the dimension is not too large and if the integrand is smooth, then the term Z^{-1} may dominate the error and then QMC-methods may be more efficient than MC-methods; see e.g. [22] and the references therein.

The efficiency of MC-methods depends on the standard deviation σ_ψ and the efficiency of QMC-methods depends on the first partial derivatives of the integrand. Higher-order smoothness is not exploited. In contrast to this, the efficiency of usual quadrature formulas depends strongly on the smoothness of the integrand.

Quadrature formulas for the high-dimensional integral Eq. (3.42) may be constructed as tensor products of one-dimensional quadrature formulas. Here, tensor products of Gauss formulas Q_ℓ with $\ell \in \mathbb{N}$ integration points $\omega^{(\ell,j)} \in \mathbb{R}$ and weights $w_{\ell,j},\ j = 1,\dots,\ell$ are used. These integrate polynomials of degree less than 2ℓ exactly and yield an error of order $O(\ell^{-(2r-1)})$ for r-times continuously differentiable integrands. For example, if the θ_i are Gaussian, then Gauss-Hermite quadrature is used. If the θ_i are uniformly distributed, then Gauss-Legendre formulas are employed.

The evaluation of Eq. (3.42) may be performed by the "full" tensor product of

the one-dimensional formulas

$$Q_l^m \psi := (Q_l^1 \otimes \cdots \otimes Q_l^1)\psi = \sum_{j_1=1}^{l} \cdots \sum_{j_m=1}^{l} w_{l,j_1} \cdots w_{l,j_m} \psi(\omega^{(l,j_1)}, \ldots, \omega^{(l,j_m)}). \tag{3.33}$$

This "full" tensor-product formula evaluates the integrand on a regular mesh of $Z = l^m$ points and yields an error of order $O(Z^{-(2r-1)/m})$. The exponential growth of the effort with dimensions makes the application to high-dimensional problems impractical. This has been termed the "curse of dimensions" [e.g. 122, 124].

Smolyak quadrature and "sparse" quadrature [163] may be applied in much higher dimensions—for recent works see [e.g. 123, 125, 137] and the references therein. A software package is also available [138].

Smolyak and sparse quadrature are discussed in detail in the Appendix B.4. Here, let us only note briefly that Smolyak quadrature formulas are constructed as sums of tensor products of one-dimensional quadrature formulas. Each term in the sum is constructed such that quadrature formulas of high order are used in only few dimensions, while formulas of low order are used in the other dimensions. For $\ell \in \mathbb{N}$ the Smolyak quadrature formula S_ℓ^m may be written as in Eq. (B.11) as

$$S_\ell^m := \sum_{\boldsymbol{\ell} \in \mathbb{N}^m :\, \ell \le |\boldsymbol{\ell}| \le \ell+m-1} (-1)^{\ell+m-1-|\boldsymbol{\ell}|} \binom{\ell-1}{|\boldsymbol{\ell}|-\ell} \cdot Q_{\ell_1} \otimes \cdots \otimes Q_{\ell_m}, \tag{3.34}$$

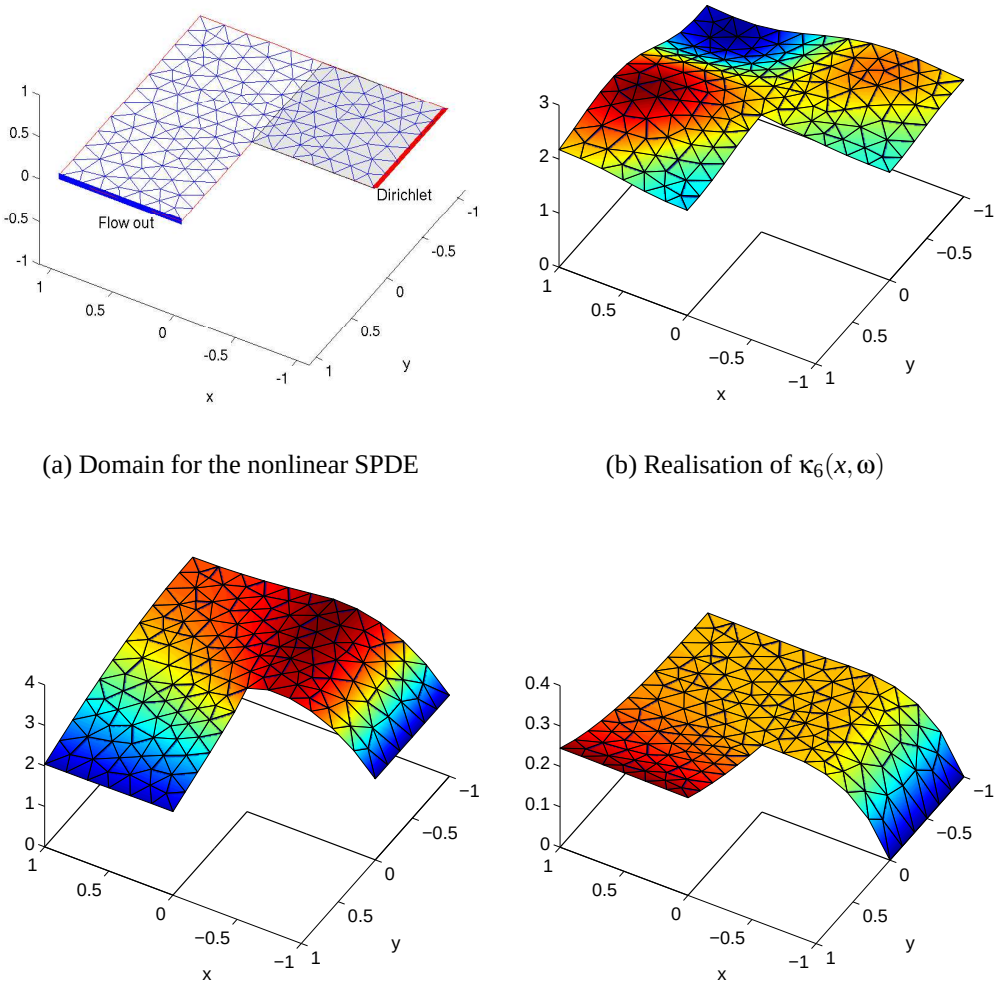
where $|\boldsymbol{\ell}| = \sum_{i=1}^m \ell_i$ for $\boldsymbol{\ell} \in \mathbb{N}^m$.

For a fixed ℓ, the number of evaluations grows significantly slower with dimensions than for full quadrature. The price is a larger error: full quadrature Q_ℓ^m integrates polynomials $\omega_1^{k_1} \cdots \omega_N^{k_m}$ exactly if their partial polynomial degree $\max_i k_i$ does not exceed $2\ell - 1$. Smolyak formulas S_ℓ^m integrate multivariate polynomials exactly only if their total polynomial degree $\sum_{i=1}^m k_i$ is at most $2\ell - 1$.

As in [125], Gauss formulas chosen according to the probability distributions of the θ_i are used in the numerical examples of this thesis to construct Smolyak quadrature formulas. Their advantage is a high exactness for smooth integrands. However, usually nested integration formulas are used in the literature, i.e. formulas where the integration points of the low-order formulas are subsets of the integration points of the high-order formulas. The integration points of such formulas, e.g. of certain Clenshaw-Curtis formulas, form a sparse grid for which the number of integration points grows slowly with dimensions; this is discussed in detail in Appendix B.4 or in [139].

3.3.4 Numerical Tests

As a numerical example (taken from [84]), the mean and the standard deviation for a nonlinear groundwater flow problem similar to Eq. (2.20) were computed by



(a) Domain for the nonlinear SPDE

(b) Realisation of $\kappa_6(x,\omega)$

(c) Mean of Reference solution

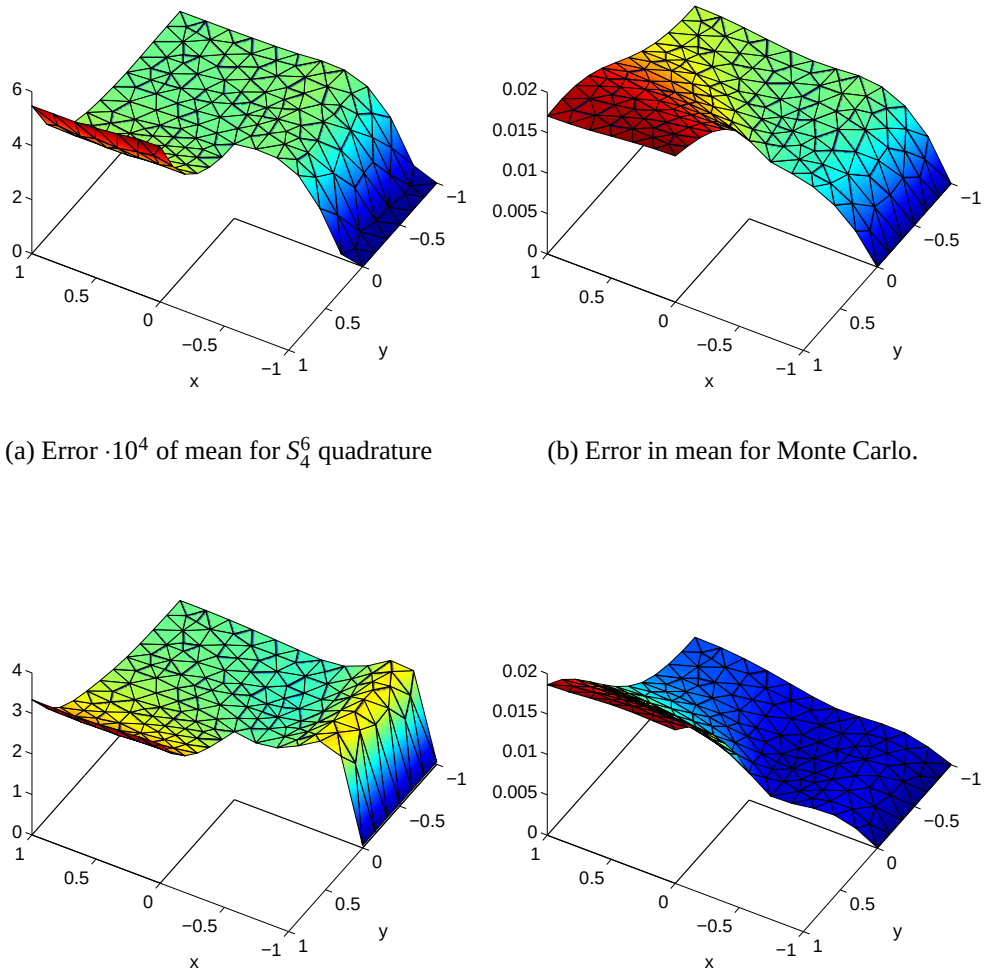
(d) Std. deviation of reference solution.

Figure 3.3: Nonlinear SPDE: Geometry and Material Realisation

naive Monte Carlo integration and by Smolyak quadrature.

The solved SPDE is $-\nabla_x \cdot (\hat{\kappa}(x,u(x,\omega),\omega)\nabla_x u(x,\omega)) = f(x,\omega), x \in R$, with $R \subset \mathbb{R}^2$ shown in Fig. 3.3(a). No-flow conditions were chosen as boundary conditions, except on the red and blue boundary, where Dirichlet-conditions and a flow-out condition were applied. The right hand side was chosen zero except on the gray part of the domain, where sources were defined.

The material was chosen as $\hat{\kappa}(x,u(x,\omega),\omega) = \kappa(x,\omega) + c(x,\omega)\,u(x,\omega)^2$ (cf.



(a) Error $\cdot 10^4$ of mean for S_4^6 quadrature

(b) Error in mean for Monte Carlo.

(c) Error $\cdot 10^3$ of standard deviation for S_4^6

(d) Error in std. deviation for Monte Carlo

Figure 3.4: Errors for the Solution by direct integration

Eq. (2.21)), where the soil parameter $\kappa(x,\omega)$ was chosen beta-distributed with mean $\mu_\kappa(x) = 2$ and with standard deviation $\sigma_\kappa(x) = 0.43$.

Six KL-terms of the underlying Gaussian field were kept; see Fig. 3.3(b) for a realisation. The mean and standard deviation of the reference solution shown in Fig. 3.3(c,d) were computed by Smolyak quadrature S_6^6 in altogether $Z = 6188$ integration points.

Smolyak quadrature S_4^6 with 451 integration points and Monte Carlo with 500

integration points were used for the solution. The resulting errors in the mean and in the standard deviation with respect to the reference solution are shown in Fig. 3.4. The errors for the naive Monte Carlo simulation are significantly larger than for the Smolyak integration—about forty times larger for the mean and six times larger for the standard deviation. Thus, a naive Monte Carlo simulation would require an approximately 1600 times higher effort for the same accuracy.

These experiments indicate that Smolyak quadrature may be an efficient alternative to Monte Carlo methods in stochastic mechanics, even though it was only compared to a naive Monte Carlo methods without variance reduction. Smolyak and sparse quadrature certainly merit more attention, especially as they can easily be integrated in existing Monte Carlo codes.

3.4 Series Expansions in the Stochastic Dimensions

The computation of statistics by direct integration may be expensive and hence various other techniques have been devised for the stochastic discretisation. The following text concentrates on techniques that obtain the solution by a series expansion in stochastic functions.

According to Section 2.3, solutions of SPDEs are elements of tensor product spaces $V \otimes (S)$, where V is the space of admissible functions on the spatial domain $R \subset \mathbb{R}^d$ and where (S) is the space of admissible stochastic functions; for example, $V = \mathring{H}^1(R)$ and $(S) = L^2(\Omega, \Sigma(\Theta), P)$ for the linear SPDE Eq. (2.13).

In Section 3.2, a semi-discretisation of the SPDE was obtained by an ansatz $u^h \in V^h \otimes (S)$, where $V^h \subset V$ is a finite-dimensional vector space of admissible spatial functions. The stochastic discretisation may be performed similarly by an ansatz $u^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}}$, where $\mathcal{I}$ is a finite set of indices identifying a basis of the finite dimensional ansatz-space $(S)^{\mathcal{I}} \subset (S)$. Function spaces $(S)^{\mathcal{I}}$ for the stochastic discretisation will be discussed in Section 3.4.1 with a focus on spaces of multivariate polynomials, the so-called *Wiener polynomial chaos*.

To obtain the solution $u^{h,\mathcal{I}}$, its coefficients in a basis of the ansatz space $V^h \otimes (S)^{\mathcal{I}}$ need to be computed. Various discretisation techniques were devised for this, e.g. certain response surface techniques [e.g. 88], Neumann expansion methods [8, 57, 132, 186], perturbation methods [e.g. 90], stochastic Galerkin techniques [8, 12, 38, 57, 84, 97, 113, 159, 176], and techniques based on orthogonal projections [58, 84, 113]. The review [80] gives an overview of these techniques and interprets all of them as series expansions.

Here, orthogonal projections (see Section 3.4.2) and stochastic Galerkin methods will be used (see Section 3.4.3) for the discretisation.

3.4.1 Polynomial Chaos Ansatz Spaces

In principle, any finite dimensional space $(S)^{\mathcal{I}} \subset (S)$ may be used for the stochastic ansatz, but the stochastic discretisation is challenging due to the high dimensions. Ghanem and Spanos [57] and many later works [e.g. 181] have chosen approximations in multivariate polynomials. Other works [e.g. 12, 34] have chosen $(S)^{\mathcal{I}}$ as a space of tensor products of piecewise polynomials, the supports of which form a regular mesh. Such generalisations of usual finite element spaces to stochastic ansatz spaces have been successfully used in $m = 10$ dimensions [38]. However, the vector-space dimension of piecewise polynomials on regular meshes grows exponentially with the stochastic dimensions, which makes their application impractical in much higher dimensions. A way to realise piecewise polynomial ansatz spaces in high dimensions may be sparse tensor product ansatz

spaces [61, 159], but the author is not aware of application to series expansions for SPDEs. See the review [80] for a more detailed discussion of ansatz spaces.

Expansions in global multivariate polynomials have found much attention in the literature and this thesis will focus on such expansions. The stochastic ansatz $(S)^{\mathcal{I}}$ is chosen as a space of multivariate polynomials in the mutually independent random variables $\theta = (\theta_1, \theta_2, \ldots)$; see Section 3.1.5. Consequently, if the $\theta_1, \theta_2, \ldots$ are Gaussian, then $(S)^{\mathcal{I}}$ is a subspace of Wiener's polynomial chaos [180], which is discussed in detail in Appendix A.2.3.

If some of the $\theta_1, \theta_2, \ldots$ are non-Gaussian, then the space of multivariate polynomials is sometimes called the generalised polynomial chaos [181, 182, 185]. Nonetheless, here all spaces of multivariate polynomials in independent random variables are denoted as polynomial chaos subspaces.

Stochastic ansatz spaces of polynomials are discussed in Appendix A.2.3: Throughout, $\alpha, \beta, \iota \in (\mathbb{N}_0)_c^{\mathbb{N}}$ will denote multi-indices (cf. the notational conventions in Appendix D), where the set $(\mathbb{N}_0)_c^{\mathbb{N}}$ is defined in Eq. (A.10). If the $\theta_1, \theta_2, \ldots$ are Gaussian, then an orthogonal basis of the separable space $(S) = L^2(\Omega, \Sigma(\theta), P)$ is given by the set $\{H_\alpha(\theta)\}_{\alpha \in (\mathbb{N}_0)_c^{\mathbb{N}}}$, where $H_\alpha(\theta)$ is defined in Eq. (A.11) as a product of Hermite-polynomials in the mutually independent Gaussian random variables $\theta = (\theta_1, \theta_2, \ldots)$.

Analogous results hold if some or all of the $\theta_1, \theta_2, \ldots$ are non-Gaussian (see [181, 182, 185] and the references therein. It will hence be assumed that the $H_\alpha(\theta)$ are multivariate polynomials in the independent random variables $\theta = (\theta_1, \theta_2, \ldots)$ and that they are an orthogonal basis of $(S) = L^2(\Omega, \Sigma(\theta), P)$.

The subspace $(S)^{\mathcal{I}} \subset (S)$ for the stochastic discretisation will be identified by a finite set $\mathcal{I} \subset (\mathbb{N}_0)_c^{\mathbb{N}}$ as $(S)^{\mathcal{I}} := \{H_\alpha(\theta) | \alpha \in \mathcal{I}\} \subset (S)$. The stochastic discretisation is performed by expanding the random coefficient vector of the spatial semi-discretisation Eq. (3.25) as

$$\boldsymbol{u}(\theta) \approx \boldsymbol{u}^{\mathcal{I}}(\theta) := \sum_{\alpha \in \mathcal{I}} \boldsymbol{u}^{(\alpha)} H_\alpha(\theta) \in ((S)^{\mathcal{I}})^n. \tag{3.35}$$

Each coefficient $\boldsymbol{u}^{(\alpha)} = \left(u_1^{(\alpha)}, \ldots, u_n^{(\alpha)}\right)^T$ in this expansion is a nodal vector for the finite element ansatz and all unknowns will be collected in the block-vector $\mathbf{u} = (\ldots; \boldsymbol{u}^{(\alpha)}; \ldots)$. Together with the spatial ansatz Eq. (3.23), this yields the tensor product ansatz

$$u^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}}, \tag{3.36}$$

$$u^{h,\mathcal{I}}(x, \theta) = \sum_{i=1,\ldots,n} \sum_{\alpha \in \mathcal{I}} N_i(x) H_\alpha(\theta) u_i^{(\alpha)} = \sum_{\alpha \in \mathcal{I}} \boldsymbol{N}(x) H_\alpha(\theta) \boldsymbol{u}^{(\alpha)}. \tag{3.37}$$

Note that the set $\mathcal{I}$ is finite and that each $H_\alpha(\theta)$ depends on only a finite number of the random variables $\theta_1, \theta_2, \ldots$ by definition. Hence, $\boldsymbol{u}^{\mathcal{I}}(\theta)$ also depends

on only a finite set $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ of the independent random variables. If this needs to be emphasised, $\boldsymbol{u}^{\mathcal{I}}(\boldsymbol{\theta})$ will be written instead of $\boldsymbol{u}^{\mathcal{I}}(\theta)$.

Once the block-vector $\mathbf{u}$ of coefficients is computed, statistics of $u^{h,\mathcal{I}}(x,\boldsymbol{\theta})$ may be evaluated either analytically or by the integration methods discussed in Section 3.3 and in Appendix B. As realisations of $u^{h,\mathcal{I}}(x,\boldsymbol{\theta})$ may be obtained at negligible costs, the application of numerical integration techniques is cheap.

The stochastic ansatz space $(S)^{\mathcal{I}}$ will usually be chosen as the m-dimensional polynomial chaos of degree k. This is the vector space of all multivariate polynomials in the m random variables $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ with total degree not exceeding k (the total degree of a monomial $\prod_i \theta_i^{\alpha_i}$ is equal to $\sum_i \alpha_i$). An orthogonal basis of this space is given by the set of all $H_\alpha(\boldsymbol{\theta})$ with $\alpha \in (\mathbb{N}_0)^m_{\leq k}$, where $(\mathbb{N}_0)^m_{\leq k}$ is defined in Eq. (A.12) as

$$(\mathbb{N}_0)^m_{\leq k} := \{\alpha \in \mathbb{N}_0^m \,|\, \sum_i \alpha_i \leq k\}. \tag{3.38}$$

According to Eq. (A.13), the vector-space dimension of the m-dimensional polynomial chaos of degree k is

$$|(\mathbb{N}_0)^m_{\leq k}| = \binom{m+k}{k}. \tag{3.39}$$

Hence, the size of the ansatz does not grow exponentially but only polynomially with the stochastic dimensions m for a fixed k. This may therefore be a feasible ansatz space for high-dimensional problems. Still, as Table A.3 illustrates, the size of the ansatz grows fast with k and/or with stochastic dimensions m. A way to overcome this may be adaptive techniques (see Section 4.5).

3.4.2 Solution by Orthogonal Projections

With the orthogonal basis $\{H_\alpha(\theta)\}_{\alpha \in (\mathbb{N}_0)_c^{\mathbb{N}}}$ of the space of admissible stochastic functions (S), the solution may be expressed as

$$u(x,\theta) = \sum_{\alpha \in (\mathbb{N}_0)_c^{\mathbb{N}}} u^\alpha(x) H_\alpha(\theta), \tag{3.40}$$

where $u^\alpha(x) = \|H_\alpha\|_{(S)}^{-2} (u(x,\theta)H_\alpha(\theta))_{(S)}$. The coefficients in Eq. (3.35) may hence be computed for general spaces (S) of admissible stochastic functions as

$$\boldsymbol{u}^{(\alpha)} = \|H_\alpha\|_{(S)}^{-2} \, (\boldsymbol{u}(\theta), H_\alpha(\theta))_{(S)}. \tag{3.41}$$

Here, (S) is chosen as $L^2(\Omega)$, and then

$$\boldsymbol{u}^{(\alpha)} = \frac{\mathbf{E}\,(\boldsymbol{u}(\theta)H_\alpha(\theta))}{\mathbf{E}\,(H_\alpha(\theta)^2)}.$$

The coefficients in Eq. (3.41) may be evaluated as in Section 3.3 by approximating the SPDE in a finite number of independent RVs $\boldsymbol{\theta} = (\theta_1, \dots, \theta_m)$ (the stability of this approximation was discussed in Section 3.3.1) and by computing the expectations as high-dimensional integrals. Using a similar notation as in Eq. (3.42), one may write the resulting integrals as

$$\boldsymbol{u}^{(\alpha)} \approx \|H_\alpha\|^{-2} \int_{\Omega_1} \cdots \int_{\Omega_m} \boldsymbol{u}(\boldsymbol{\theta}) H_\alpha(\boldsymbol{\theta})\, dP_{\theta_1}(\omega_1) \cdots dP_{\theta_m}(\omega_m). \tag{3.42}$$

These integrals may be evaluated by any technique discussed in Section 3.3.3 or in Appendix B, resulting in an approximation of Section 3.42 by a sum

$$\boldsymbol{u}^{(\alpha)} \approx \|H_\alpha\|^{-2} \sum_{z=1}^{Z} w_z \boldsymbol{u}(\boldsymbol{\omega}^{(z)}) H_\alpha(\boldsymbol{\omega}^{(z)}) \tag{3.43}$$

over Z integration points $\boldsymbol{\omega}^{(1)}, \dots, \boldsymbol{\omega}^{(Z)} \in \Omega^{(m)}$ (see Eq. (3.32) for the notation). For every integration point the corresponding realisation of $\boldsymbol{u}(\boldsymbol{\omega}^{(z)})$ needs to be computed, which requires to solve the corresponding realisation of the finite-dimensional approximation of the SPDE.

Thus, the block-vector $\mathbf{u}$ is obtained here by solving many uncoupled problems. This is similar to the original Monte Carlo ideas, but in contrast to these, a response surface is obtained directly. Ghiocel and Ghanem [58] name this technique the *non-intrusive stochastic finite element method* and compute the coefficients by Monte Carlo integration, whereas in [84] methods based on Smolyak integration are used. For numerical experiments see Section 4.4.3.

3.4.3 Galerkin Methods in the Stochastic Dimensions

Just as in usual finite element methods, Galerkin conditions may also be imposed in the stochastic dimensions to compute the coefficients $\mathbf{u}$ of the expansion. This was first proposed for stochastic finite elements by Ghanem and Spanos [57].

By inserting the tensor product ansatz Eq. (3.37) into the SPDE and by applying Galerkin conditions, a discretisation is obtained that requires to

$$\begin{aligned} &\text{find } u^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}} \text{ such that} \\ &a(u^{h,\mathcal{I}}, v^{h,\mathcal{I}}) = f(v^{h,\mathcal{I}}) \qquad \text{for all } v^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}}. \end{aligned} \tag{3.44}$$

Here, a is the bilinear form defined in Eq. (2.17) or the semilinear form defined in Eq. (2.25). With the basis $\{N_i(x) H_\alpha(\theta)\}_{i,\alpha}$ of $V^h \otimes (S)^{\mathcal{I}}$ and the spatial semi-discretisation Eq. (3.26), Eq. (3.44) may be written as the system of the $n \cdot |\mathcal{I}|$ coupled equations

$$\boldsymbol{r}^{(\beta)}(\mathbf{u}) := \mathbf{E}\left(\boldsymbol{r}\Big(\sum_{\alpha \in \mathcal{I}} \boldsymbol{u}^{(\alpha)} H_\alpha(\theta), \theta \Big) H_\beta(\theta) \right) = 0, \quad \forall \beta \in \mathcal{I}. \tag{3.45}$$

The residual may be written as the block-vector $\mathbf{r}(\mathbf{u}) = (\ldots, \boldsymbol{r}^{(\beta)}(\mathbf{u}), \ldots)^T$, where each sub-vector $\boldsymbol{r}^{(\beta)}(\mathbf{u})$ is a coefficient vector for the spatial ansatz. Numerical techniques for the solution are discussed in the next chapter.

In contrast to the direct integration techniques of Section 3.3 this discretisation does not require an explicit approximation of the SPDE in a finite number of independent random variables. Instead, a finite-dimensional representation of the solution is obtained by the choice of the stochastic ansatz and by the Galerkin projection. In contrast to the other discretisation techniques discussed here, the Galerkin method does not require to perturb the SPDE. For linear SPDEs, even the practical implementation of the discretisation can be performed without approximating the random fields; see Section 4.1.1.

The convergence of the Galerkin approximation follows from the usual theory:

Theorem 3.2: *Denote by u the solution of the linear elliptic SPDE Eq. (2.17) or of the nonlinear elliptic SPDE Eq. (2.25), and denote by $u^{h,\mathcal{I}}$ the respective Galerkin-approximation obtained in Eq. (3.44). Both in the linear and in the nonlinear case, there exists a constant C independent of h and $\mathcal{I}$ such that*

$$\|u - u^{h,\mathcal{I}}\|_{V\otimes(S)} \leq C \inf_{v^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}}} \|u - v^{h,\mathcal{I}}\|_{V\otimes(S)}.$$

PROOF: For the linear SPDE this follows from Cèa's Lemma [27, Theorem 2.4.1, p. 104]. In the nonlinear case this follows from a generalisation of Cèa's Lemma for monotone nonlinear operators [27, Theorem 5.3.4, p. 322]. □

For linear SPDEs, the residual Eq. (3.45) may be written as

$$\boldsymbol{r}^{(\beta)}(\mathbf{u}) = \sum_{\alpha\in\mathcal{I}} \underbrace{\mathbf{E}\left(\boldsymbol{K}(\theta)H_\alpha(\theta)H_\beta(\theta)\right)}_{=:\boldsymbol{K}_{\alpha,\beta}} \boldsymbol{u}^{(\alpha)} - \underbrace{\mathbf{E}\left(\boldsymbol{f}(\theta)H_\beta(\theta)\right)}_{=:\boldsymbol{f}^{(\beta)}}, \qquad \beta \in \mathcal{I}. \quad (3.46)$$

With the block matrix $\mathbf{K} = (\boldsymbol{K}_{\alpha,\beta})_{\alpha,\beta}$ and with the block vector $\mathbf{f} = (\ldots; \boldsymbol{f}^{(\beta)}; \ldots)$, which are defined as

$$\boldsymbol{K}_{\alpha,\beta} = \mathbf{E}\left(\boldsymbol{K}(\theta)H_\alpha(\theta)H_\beta(\theta)\right), \qquad \alpha,\beta \in \mathcal{I}, \quad (3.47)$$

$$\boldsymbol{f}^{(\beta)} = \mathbf{E}\left(\boldsymbol{f}(\theta)H_\beta(\theta)\right), \qquad \beta \in \mathcal{I}, \quad (3.48)$$

this may be written as the linear block system with $n \cdot |\mathcal{I}|$ equations

$$\mathbf{K}\mathbf{u} = \mathbf{f}. \quad (3.49)$$

Theorem 3.3: *The block-matrix $\mathbf{K}$ in Eq. (3.49) is symmetric iff the bilinear form in Eq. (2.17), Eq. (3.44) is symmetric. It is positive definite if this bilinear is coercive.*

PROOF: Eq. (3.49) is the representation of Eq. (3.44) in a basis of $V^h \otimes (S)^{\mathcal{I}}$. The theorem holds as a is a symmetric elliptic bilinear form. □

Example 3.3: As an example, the discretisation of the linear SPDE Eq. (2.13) with the spatial discretisation of Example 3.2 may be written as

$$\boldsymbol{K}_{\alpha,\beta} = \int_R \nabla \boldsymbol{N}(x)\, \mathbf{E}\left(\kappa(x,\theta) H_\alpha(\theta) H_\beta(\theta)\right) \left(\nabla \boldsymbol{N}(x)\right)^T dx \qquad \text{and} \qquad (3.50)$$

$$\boldsymbol{f}^{(\beta)} = \int_R \mathbf{E}\left(f(x,\theta) H_\beta(\theta)\right) \left(\boldsymbol{N}(x)\right)^T dx. \qquad (3.51)$$

The discretisation of the nonlinear SPDE Eq. (2.20) with the spatial discretisation of Example 3.2 may be written for $\beta \in \mathcal{I}$ as

$$\boldsymbol{r}^{(\beta)}(\mathbf{u}) = \mathbf{E}\left(\int_R \nabla \boldsymbol{N}(x) G\Big(\sum_{\alpha\in\mathcal{I}} \boldsymbol{N}(x) H_\alpha(\theta) \boldsymbol{u}^{(\alpha)}\Big)\, dx\, H_\beta(\theta)\right) - \boldsymbol{f}^{(\beta)},$$

where $\boldsymbol{f}^{(\beta)}$ is defined as in Eq. (3.51).

3.4.4 Convergence Rates

Convergence rates for the stochastic Galerkin method are discussed now. According to Cèa's Lemma 3.2, the error of the Galerkin method is $\|u - u^{h,\mathcal{I}}\| \leq C \inf_{v^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}}} \|u - v^{h,\mathcal{I}}\|_{V\otimes(S)}$.

À priori errors for the stochastic Galerkin methods may hence be obtained if it is known how well u is approximated by the ansatz space. Two ingredients are required for this: the regularity of u needs to be known, where the regularity is expressed by specifying a suitable subspace of $V \otimes (S)$ in which u is contained. Additionally, the error in polynomial chaos approximations for functions from this space must be specified. This is analogous to the usual finite element theory [27, 65, 127, 165], where à priori errors are obtained by combining Cèa's Lemma with approximation results for piecewise polynomial interpolations in Sobolev spaces.

Two convergence results are reviewed below. The first holds only if the SPDE depends on a finite number of independent random variables, whereas the second also holds for general SPDEs.

Finite-Dimensional Probability Spaces: À priori convergence results for linear SPDEs depending on a finite number of independent random variables were given by Babuška, Tempone and Zouaris [12] who assumed the solution to be smooth in the stochastic parameters.

This assumption of smoothness is often justifiable if the SPDE depends on only a finite number of independent random variables: consider the SPDE

$$-\nabla\cdot(\kappa(x,\boldsymbol{\theta})\nabla u(x,\boldsymbol{\theta})) = f(x,\boldsymbol{\theta}), \tag{3.52}$$

which depends on only a finite number of mutually independent random variables $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_m)$. The derivative of u w.r.t. the random variable θ_i satisfies the elliptic SPDE $-\nabla\cdot(\kappa(x,\boldsymbol{\theta})\nabla(\partial_{\theta_i}u(x,\boldsymbol{\theta}))) = \partial_{\theta_i}f(x,\boldsymbol{\theta}) + \nabla\cdot((\partial_{\theta_i}\kappa(x,\boldsymbol{\theta}))\nabla u(x,\boldsymbol{\theta}))$. Higher-order derivatives may be computed similarly. Thus, if κ is smooth in $\boldsymbol{\theta}$, then $u(x,\boldsymbol{\theta})$ is also smooth in $\boldsymbol{\theta}$.

This smoothness is exploited in [12] to state convergence rates for the stochastic Galerkin-method. Assume that the stochastic ansatz consists of all multivariate polynomials $H_\beta(\boldsymbol{\theta})$ with $0 \le \beta_i \le p_i, i = 1,\ldots,m$. Denote by $u^{h,\boldsymbol{p}}$ the Galerkin approximation in the tensor product of this stochastic ansatz space with a spatial ansatz space of piecewise linear polynomials on triangular finite elements with maximum diameter $h > 0$. With constants $c, c_1, \ldots, c_m > 0$, the following result holds [12, Theorem 6.3]:

$$\|u - u^{h,\boldsymbol{p}}\|_{L^2(R)\otimes L^1(\Omega)} \le c\Big(h^2 + \sum_{i=1}^{m}(c_i)^{2p_i+2}\Big). \tag{3.53}$$

This theorem gives à priori convergence rates for fixed m, but convergence cannot be concluded from Eq. (3.53) for $m \to \infty$.

General Probability Spaces: For general probability spaces, the smoothness of u in the basic independent random variables is not an appropriate concept for characterising the stochastic regularity: On one hand, an infinite-dimensional differential calculus is required in general probability spaces, e.g. the Malliavin calculus [100]. On the other hand, the classical concept of smoothness does not force the polynomial chaos coefficients $u^{(\beta)}(x)$ to decrease with growing dimensions (with growing length of β).

The stochastic regularity of random parameters is usually expressed by stating in which Hida- or Kondratiev-spaces of stochastic distributions or of stochastic test functions (see Appendix A.3) they are contained in. Variational theories for the convergence rates of stochastic finite element solutions of SPDEs in these spaces were presented by Benth and Gjerde [15] and applied by Theting [171].

Results on the error of polynomial chaos approximations for stochastic distributions from the space $(S)^{-\rho,-q}$ with $\rho \in (0,1]$ and $q \in \mathbb{R}$ (see Appendix A.3 for the definition) were given in [15]. These results are summarised in Appendix A.3, and the estimate in Eq. (A.23) can immediately be used to obtain à priori errors for the stochastic Galerkin method.

Assume that the solution u of the linear elliptic SPDE Eq. (2.18) is an element of the space $\big(\mathring{H}^1(R)\cap H^2(R)\big)\otimes(S)^{-\rho,-q}$. Let $u^{h,m}_{\le k}$ be the approximation

obtained by the Galerkin method of Section 3.4, where the spatial discretisation is performed by triangular finite elements with maximum diameter $h > 0$ with piecewise linear ansatz functions and where the m-dimensional polynomial chaos of degree k is used for the stochastic ansatz. Choose $r \geq r^*$ as in Theorem A.2 ($r^* \approx 1.53$) and let $p = r + q$. Then (cf. Eq. (A.23), [15, Theorem 5.1], [171, Theorem 6.1]):

$$\|u - u^{h,m}_{\leq k}\|_{H^1 \otimes (S)^{-\rho,-p}} \leq C\Big(h\,\|u\|_{H^2 \otimes (S)^{-\rho,-p}} + c(m,k,p-q)\|u\|_{H^1 \otimes (S)^{-\rho,-q}}\Big), \tag{3.54}$$

where $H^1 = H^1(R)$ and $H^2 = H^2(R)$ denote the usual Sobolev spaces, where $c(m,k,r)$ is defined in Eq. (A.22) as

$$c(m,k,r) = \sqrt{c_1(r)m^{1-r} + c_2(r)\Big(\frac{r}{2^r(r-1)}\Big)^{k+1}}. \tag{3.55}$$

Note that Galerkin solutions of Wick SPDEs are considered in [15, 171]. Nonetheless, the above results are also valid for the SPDEs considered here, because the à priori error Eq. (3.54) depends only on the assumption $u \in \big(\mathring{H}^1(R) \cap H^2(R)\big) \otimes (S)^{-\rho,-q}$ and on the approximation properties of the polynomial chaos spaces.

The estimate Eq. (3.54) yields a useful estimate for $m \to \infty$ as $c(m,k,r) \to 0$ for $m \to \infty$. However, these requirements are different from these in Eq. (3.53). To apply this estimate requires information about the regularity of the solution and a theory for this still needs to be devised.

Chapter 4

Numerical Solution of Stochastic Partial Differential Equations

This chapter discusses solvers for the large system of equations Eq. (3.44) obtained by the polynomial chaos Galerkin discretisation.

Solvers for linear SPDEs are discussed in sections 4.1–4.3. The tensor product structure of the Galerkin discretisation leads to a large system of block equations $\mathbf{Ku} = \mathbf{f}$, which may be stored and solved with less effort than its size suggests. Efficient Kronecker product representations for the block matrix $\mathbf{K}$ are developed in Section 4.1. There, it is also discussed how $\mathbf{K}$ may be evaluated in practice.

These Kronecker product representations allow the efficient execution of the block matrix-vector product $\mathbf{Ku}$. It is hence advantageous to solve the linear block system by iterative solvers. Solution techniques that exploit this structure are discussed in Section 4.2. The block matrix representation developed in Section 4.1.2 further permits to parallelise the iterative solvers; this is discussed in Section 4.3.

The solution of nonlinear SPDEs is addressed in Section 4.4. This requires to evaluate the residual, which is discussed in Section 4.4.1. Iterative solvers for the resulting nonlinear block system of equations are covered in Section 4.4.2. An adaptive solver is presented in Section 4.5.

4.1 Representation of Discretised Linear Stochastic Partial Differential Equations

The discretisation of a linear elliptic SPDE by the stochastic Galerkin method of Section 3.4.3 yields Eq. (3.49), a linear system of block equations

$$\mathbf{Ku} = \mathbf{f}, \tag{4.1}$$

where the block matrix $\mathbf{K} = \left(\boldsymbol{K}_{\alpha,\beta}\right)_{\alpha,\beta\in\mathcal{I}}$ and the block vector $\mathbf{f} = (\ldots, \boldsymbol{f}^{(\beta)}, \ldots)^T$ are given by Eqs. (3.47) and (3.48). Recall that the block matrix $\mathbf{K}$ is symmetric positive definite according to Theorem 3.3.

It is now discussed how to compute and represent the block matrix $\mathbf{K}$: The linear system Eq. (4.1) comprises $n \cdot |\mathcal{I}|$ equations. As this number is usually large, storing $\mathbf{K}$ in full form would lead to high memory requirements and to high costs for the matrix-vector product with $\mathbf{K}$. Sections 4.1.1 and 4.1.2 discuss two Kronecker product representations of the block matrix $\mathbf{K}$ with small memory demands and with small costs for computing the matrix-vector product with $\mathbf{K}$.

For simplicity, the computation and representation of the system matrix $\mathbf{K}$ are developed here for the linear SPDE Eq. (2.13); the generalisation is straightforward. This SPDE, represented in the mutually independent random variables $\theta = (\theta_1, \theta_2, \ldots)$, is

$$\begin{aligned} -\nabla \cdot (\kappa(x,\theta)\nabla u(x,\theta)) &= f(x,\theta), & x &\in R \\ u(x,\theta) &= 0, & x &\in \partial R. \end{aligned} \tag{4.2}$$

For the practical evaluation of $\mathbf{K}$ it will be assumed that a code is available that discretises and solves the deterministic analogon of our SPDE. As before, this software will be called the *deterministic code* or the *deterministic solver*.

4.1.1 Polynomial Chaos Expansion of the System Matrix

As was proposed by Ghanem and his co-workers [49, 50, 52, 57, 135], the block matrix $\mathbf{K}$ may be represented efficiently by expanding the semi-discretisation of the system matrix in polynomial chaos. Recall that this semi-discretisation in the finite element space $V^h \subset V$ was given in Eq. (3.27) as

$$\boldsymbol{K}(\theta) = \int_R \nabla \boldsymbol{N}(x)\kappa(x,\theta)(\nabla \boldsymbol{N}(x))^T \, dx. \tag{4.3}$$

It was required in Section 2.3.1 that $\kappa \in L^\infty(R) \otimes L^\infty(\Omega)$. Hence, $\kappa \in L^\infty(R) \otimes L^2(\Omega)$ and thus $\boldsymbol{K}(\theta) \in (L^2(\Omega))^{n\times n}$. The semi-discretisation may therefore be written as the L^2-convergent expansion

$$\boldsymbol{K}(\theta) = \sum_{\iota\in(\mathbb{N}_0)_c^{\mathbb{N}}} \boldsymbol{K}^{(\iota)} H_\iota(\theta), \tag{4.4}$$

where $\boldsymbol{K}^{(\iota)} := \|H_\iota(\theta)\|^{-2}\,\mathbf{E}\left(\boldsymbol{K}(\theta)H_\iota(\theta)\right))$.

The matrices $\boldsymbol{K}^{(\iota)}, \iota \in (\mathbb{N}_0)_c^{\mathbb{N}}$, may be computed by calling the deterministic code with $\kappa^{(\iota)}(x) = \|H_\iota(\theta)\|^{-2}\,\mathbf{E}\left(\kappa(x,\theta)H_\iota(\theta)\right))$ as material parameter. If the deterministic code is a finite element software, this usually requires the values of $\kappa^{(\iota)}(x)$ at the integration points of the finite elements.

Note that the iterative solvers discussed later will not require that the matrices $\boldsymbol{K}^{(\iota)}$ are stored explicitly. They only require a routine that computes the matrix-vector product of $\boldsymbol{K}^{(\iota)}$ with a finite element nodal vector. Whether it is advantageous to retrieve the stiffness matrices from the deterministic code and store them, or whether it is better to call the matrix-vector product of the deterministic code depends on the problem size and on the communication costs. The following text will not always distinguish between obtaining a matrix from the deterministic code and calling the matrix-vector-product routine of the deterministic code: if the text speaks of "obtaining" a stiffness matrix, then the actual implementation may also use the matrix-vector-product.

4.1.1.1 Practical Computation of the Expansion

The projection $\kappa^{(\iota)}(x) = \|H_\iota(\theta)\|^{-2}\mathbf{E}\left(\kappa(x,\theta)H_\iota(\theta)\right)$ may be computed by evaluating the expectation operator numerically, e.g. by the techniques for high-dimensional integration discussed in Section 3.3.

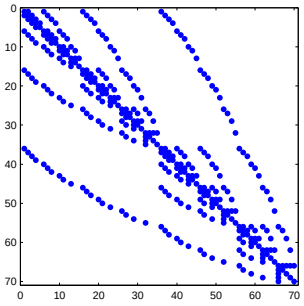
For a random field κ that is specified as a transformation $\kappa(x,\theta) = \phi(x,\gamma(x,\theta))$ of a Gaussian field γ (see Section 2.2.4.1) it is shown now how the projection may be computed analytically for certain types of marginal distributions. Denote the Karhunen–Loève expansion of γ by $\gamma(x,\theta) = \sum_{i=1}^{\infty}\sqrt{\nu_i}\gamma_i(x)\,\theta_i$. Here, the ν_i and the γ_i are the eigenvalues and the eigenfunctions of the KL-eigenvalue problem for γ, whereas the θ_i are mutually independent standard Gaussian random variables (see Section 3.1). The projection onto H_ι may be evaluated by Eq. (A.16) if the transformation $\phi(x,\gamma)$ is smooth in γ:

$$\begin{aligned}
\kappa^{(\iota)}(x) &= \|H_\iota(\theta)\|^{-2}\mathbf{E}\left(\kappa(x,\theta)H_\iota(\theta)\right) && (4.5)\\
&= \frac{1}{\iota!}\,\mathbf{E}\left(\kappa(x,\theta)H_\iota(\theta)\right) && (4.6)\\
&= \frac{1}{\iota!}\,\mathbf{E}\Big(D^\iota\phi(x,\sum_{i}^{\infty}\sqrt{\nu_i}\iota_i(x)\theta_i)\Big), \qquad \text{due to Eq. (A.16),} && (4.7)\\
&= \frac{1}{\iota!}\,\mathbf{E}\Big(\frac{\partial^{|\iota|}}{\partial\gamma^{|\iota|}}\phi(x,\gamma)|_{\gamma=\gamma(x,\omega)}\Big)\prod_{i=1}^{\infty}(\sqrt{\nu_i}\gamma_i(x))^{\iota_i}. && (4.8)
\end{aligned}$$

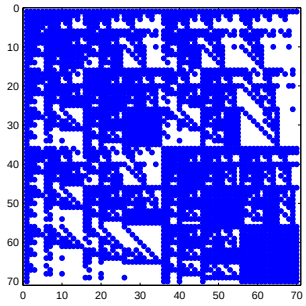
To apply this formula, one needs to evaluate the expectation in Eq. (4.8). For various marginal distributions this expression may be used with success to obtain an analytical expression for $\kappa^{(\iota)}$.

4.1.1.2 Using the Exact Expansion in Practice

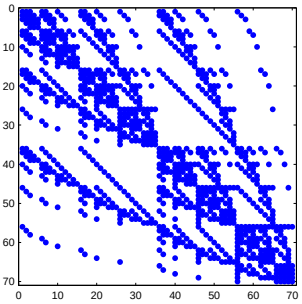
In practice, the expansion of $\boldsymbol{K}(\theta)$ needs to be truncated after a finite number of terms. The approximation used for the semi-discretisation is then $\boldsymbol{K}(\theta) \approx$



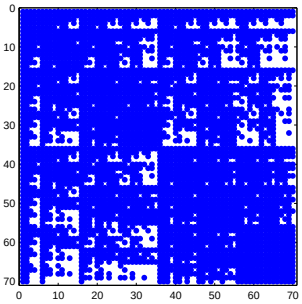
(a) Block sparsity, $\mathcal{J} = (\mathbb{N}_0)^m_{\leq 1}$



(b) Block sparsity, $\mathcal{J} = (\mathbb{N}_0)^m_{\leq 4}$



(c) Block sparsity, $\mathcal{J} = (\mathbb{N}_0)^m_{\leq 2}$



(d) Block sparsity, $\mathcal{J} = (\mathbb{N}_0)^m_{\leq 5}$

Figure 4.1: Block sparsity structure of $\mathbf{K}$ for $\mathcal{I} = (\mathbb{N}_0)^m_{\leq 4}$. Every dot denotes a non-zero matrix $\boldsymbol{K}_{\alpha,\beta}$.

$\sum_{\iota\in\mathcal{J}} \boldsymbol{K}^{(\iota)} H_\iota(\theta)$, where $\mathcal{J} \subset (\mathbb{N}_0)^{\mathbb{N}}_c$ is a finite set, which must be chosen in accordance with the stochastic ansatz $(S)^{\mathcal{I}}$. It is shown now that $\mathcal{J}$ may be chosen such that an exact representation of the system matrix $\mathbf{K}$ is obtained.

As shown in [57], the sub-matrix $\boldsymbol{K}_{\alpha,\beta}, \alpha,\beta \in \mathcal{I}$, of the system matrix $\mathbf{K}$ may be written by inserting Eq. (4.4) into the discretised form Eq. (3.50) as

$$\boldsymbol{K}_{\alpha,\beta} = \sum_{\iota\in(\mathbb{N}_0)^{\mathbb{N}}_c} \mathbf{E}\left(H_\alpha(\theta) H_\beta(\theta) H_\iota(\theta)\right) \boldsymbol{K}^{(\iota)}. \tag{4.9}$$

The following theorem [113] shows that the sum over ι is finite:

Theorem 4.1: *The sum over ι in Eq. (4.9) is finite, more specifically*

$$\mathbf{E}\left(H_\alpha(\theta) H_\beta(\theta) H_\iota(\theta)\right) = 0, \qquad \text{if } \iota_i > \alpha_i + \beta_i \text{ for all } i = 1,2,\ldots. \tag{4.10}$$

order	α	order	α	order	α
0	$\alpha=(0,0,0,0)$	5	$\alpha=(0,2,0,0)$	10	$\alpha=(0,0,0,1)$
1	$\alpha=(1,0,0,0)$	6	$\alpha=(0,0,1,0)$	11	$\alpha=(1,0,0,1)$
2	$\alpha=(2,0,0,0)$	7	$\alpha=(1,0,1,0)$	12	$\alpha=(0,1,0,1)$
3	$\alpha=(0,1,0,0)$	8	$\alpha=(0,1,1,0)$	13	$\alpha=(0,0,1,1)$
4	$\alpha=(1,1,0,0)$	9	$\alpha=(0,0,2,0)$	14	$\alpha=(0,0,0,2)$

Table 4.1: The order assigned to $\alpha \in (\mathbb{N}_0)^m_{\leq 2}$.

PROOF: The product $H_\alpha(\theta)H_\beta(\theta)$ is a multivariate polynomial in the random variables $\theta=(\theta_1,\theta_2,\ldots)$, in which θ_i appears with maximum exponent $\alpha_i+\beta_i$. Hence, $H_\alpha(\theta)H_\beta(\theta)=\sum_{\{\eta|\eta_i\leq\alpha_i+\beta_i, i\in\mathbb{N}\}} h^{(\eta)}H_\eta(\theta)$. If $\iota_i>\alpha_i+\beta_i$ for all $i\in\mathbb{N}$, then H_ι is orthogonal to all H_η in the sum by definition of the polynomial chaos (see Section A.2.3). Thus, $\mathbf{E}\left(H_\alpha H_\beta H_\iota\right)=\sum_{\{\eta|\eta_i\leq\alpha_i+\beta_i, i\in\mathbb{N}\}} h^{(\eta)}\mathbf{E}\left(H_\eta H_\iota\right)=0$. □

With the set

$$\mathcal{J}(\mathcal{I}) := \{\iota \mid \iota_i \leq \alpha_i+\beta_i \text{ for all } i \text{ and all } \alpha,\beta\in\mathcal{I}\} \tag{4.11}$$

and with the matrices

$$\Delta^{(\iota)}_{\alpha,\beta} = \mathbf{E}\left(H_\alpha(\theta)H_\beta(\theta)H_\iota(\theta)\right), \qquad \alpha,\beta\in\mathcal{I}, \tag{4.12}$$

the block matrix may be written as a sum of Kronecker products as

$$\mathbf{K} = \sum_{\iota\in\mathcal{J}(\mathcal{I})} \Delta^{(\iota)}\otimes \boldsymbol{K}^{(\iota)}. \tag{4.13}$$

In the experiments, the stochastic ansatz space is usually chosen as the polynomial chaos of degree k in m random variables. Then $\mathcal{I}=(\mathbb{N}_0)^m_{\leq k}$ (see Eq. (A.12)) and then $\mathcal{J}(\mathcal{I})=(\mathbb{N}_0)^m_{\leq 2k}$ is the polynomial chaos of twice this degree.

However, choosing $\mathcal{J}=\mathcal{J}(\mathcal{I})$ according to Theorem 4.1 results in $\mathbf{K}$ being a full block matrix ($\mathbf{K}$ still has the sparsity structure of the individual $\boldsymbol{K}^{(i)}$, i.e. of a typical FE stiffness matrix). It may therefore be advantageous to choose a smaller set $\mathcal{J}$ for the expansion of the operator. This is demonstrated in Fig. 4.1, where the block sparsity structure of $\mathbf{K}$ for different choices of $\mathcal{J}$ is shown. The stochastic ansatz is chosen as the four-dimensional polynomial chaos of degree four, hence $\mathcal{I}=(\mathbb{N}_0)^m_{\leq 4}$. According to Theorem 4.1, $\mathcal{J}$ should be chosen as $\mathcal{J}=(\mathbb{N}_0)^m_{\leq 8}$. This case is not shown in Fig. 4.1 as it leads to a full block matrix $\mathbf{K}$.

Some order needs to be introduced on the sets of multi-indices for practical computations. In the following, the multi-indices are ordered ascending from

right to left (see Table 4.1 for an example) and the set $\mathcal{I}$ is identified with the set $\{1,\dots,|\mathcal{I}|\}$.

To visualise the structure of Eq. (4.13), the system of block equations Eq. (4.1) is now written as

$$\mathbf{K}\mathbf{u} = \sum_{\iota\in\mathcal{J}(\mathcal{I})} \left[\Delta^{(\iota)} \otimes \boldsymbol{K}^{(\iota)}\right] \mathbf{u} = \mathbf{f} \tag{4.14}$$

$$= \sum_{\iota\in\mathcal{J}(\mathcal{I})} \begin{pmatrix} \Delta^{(\iota)}_{1,1}\boldsymbol{K}^{(\iota)} & \cdots & \Delta^{(\iota)}_{1,|\mathcal{I}|}\boldsymbol{K}^{(\iota)} \\ \vdots & \ddots & \vdots \\ \Delta^{(\iota)}_{|\mathcal{I}|,1}\boldsymbol{K}^{(\iota)} & \cdots & \Delta^{(\iota)}_{|\mathcal{I}|,|\mathcal{I}|}\boldsymbol{K}^{(\iota)} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{(1)} \\ \vdots \\ \boldsymbol{u}^{(|\mathcal{I}|)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^{(1)} \\ \vdots \\ \boldsymbol{f}^{(|\mathcal{I}|)} \end{pmatrix}. \tag{4.15}$$

(4.16)

The iterative solvers discussed later will require block matrix-vector products with $\mathbf{K}$. It is hence not necessary to store $\mathbf{K}$. Nonetheless, storing the matrix may be advantageous for efficiency reasons, e.g. if the communication costs with the deterministic code are high. The following example shows that the Kronecker product representation reduces the memory requirements and demonstrates that a naive implementation of the block matrix-vector product increases the effort considerably compared to a full representation.

Example 4.1: Assume that the stochastic ansatz is the 10-dimensional polynomial chaos of degree 4. Then $|\mathcal{I}| = |(\mathbb{N}_0)^m_{\leq 4}| = 1001$. Due to symmetry, to store $\mathbf{K}$ in full form requires to retrieve about half a million stiffness matrices $\boldsymbol{K}_{\alpha,\beta}, \alpha,\beta \in \mathcal{I}$, from the deterministic code and to store them. The block matrix-vector product with $\mathbf{K}$ needs to perform approximately half a million matrix-vector products with the finite element stiffness matrices $\boldsymbol{K}_{\alpha,\beta}$.

To store $\mathbf{K}$ in the Kronecker product representation Eq. (4.14) requires to retrieve only $|\mathcal{J}(\mathcal{I})| = |\mathcal{I}_{10,8}| = 43,758$ stiffness matrices $\boldsymbol{K}^{(\iota)}$ from the deterministic code and to store them. A direct implementation of the block matrix-vector product as $(\mathbf{K}\mathbf{u})_\alpha = \sum_\iota \sum_\beta \Delta^{(\iota)}_{\alpha,\beta}\boldsymbol{K}^{(\iota)}\boldsymbol{u}^{(\beta)}$ requires $|\mathcal{J}(\mathcal{I})| \cdot |\mathcal{I}| \approx 44$ million matrix-vector products $\mathbf{K}^{(\iota)}\boldsymbol{u}^{(\beta)}$. This is a much larger number than before and the block matrix-vector product will hence be implemented differently.

The block matrix-vector product may be implemented more efficiently by exploiting the block sparsity pattern of the matrix $\mathbf{K}$ shown in Fig. 4.1. This pattern may be computed à priori by storing the non-zero elements $\mathbf{E}\left(H_\alpha H_\beta H_\iota\right)$ in a sorted list. Each $\boldsymbol{K}_{\alpha,\beta}$ may then be computed from Eq. (4.9) and the block matrix-vector product may be performed using the thus computed matrices $\boldsymbol{K}_{\alpha,\beta}$.

The number of matrix-vector products used in the block matrix-vector product does then not exceed the number that was originally required. If the set $\mathcal{J}$ is chosen according to Theorem 4.1, then the number of required matrix-vector products

is the same as for the full representation. But if a smaller set $\mathcal{J}$ is chosen, then the known block sparsity may be exploited and less matrix-vector products are needed.

4.1.2 Karhunen–Loève Expansion of the System Matrix

A representation that is more advantageous than the one discussed in the previous section is obtained by expanding the material parameter in its Karhunen–Loève expansion. As noted in the previous section, $\kappa \in L^\infty(R) \otimes L^2(\Omega, \Sigma(\theta), P)$. Hence, its KL-series $\kappa(x,\theta) = \mu_\kappa(x) + \sum_{i=1}^\infty \sqrt{\lambda_i}\kappa_i(x)\xi_i(\theta)$ is $L^2(R) \otimes L^2(\Omega)$-convergent. Here, λ_i and κ_i are the eigenvalues and eigenfunctions of the KL-eigenvalue problem for the covariance function cov_κ and the $\xi_i(\theta)$ are uncorrelated centred random variables with unit variance (see Section 3.1). For convenience, define $\lambda_0 := 1$, $\theta_0(\theta) := 1$, and $\kappa_0(x) := \mu_\kappa(x)$.

As $\xi_i \in L^2(\Omega, \Sigma(\theta), P), i = 0, 1, \ldots$, the polynomial chaos expansion $\xi_i(\theta) = \sum_\iota \xi_i^{(\iota)} H_\iota(\theta)$ is $L^2(\Omega)$-convergent. Therefore the expansion

$$\kappa(x,\theta) = \sum_{i=0}^\infty \sum_{\iota \in (\mathbb{N}_0)_c^\mathbb{N}} \sqrt{\lambda_i}\xi_i^{(\iota)}\kappa_i(x)H_\iota(\theta) \tag{4.17}$$

is $L^2(R) \otimes L^2(\Omega)$-convergent. Hence, the spatial semi-discretisation of the system matrix for the example Eq. (4.2) has the $(L^2(\Omega))^{n\times n}$-convergent expansion

$$\boldsymbol{K}(\theta) = \sum_{i=0}^\infty \sum_{\iota \in (\mathbb{N}_0)_c^\mathbb{N}} \sqrt{\lambda_i}\xi_i^{(\iota)} \boldsymbol{K}_i H_\iota(\theta). \tag{4.18}$$

Here, $\boldsymbol{K}_i = \int_R \nabla \boldsymbol{N}(x)\kappa_i(x)(\nabla \boldsymbol{N}(x))^T \, dx$ for $i = 0, 1, 2, \ldots$ (note that $\boldsymbol{K}_0$ is the mean system matrix).

As before, the matrices $\boldsymbol{K}_i$ may be computed by calling the deterministic code with $\kappa_i(x)$ as material parameter. Again, the iterative solvers discussed later will not require to store the matrices $\boldsymbol{K}_i$. Only a routine that computes matrix-vector products of the $\boldsymbol{K}_i$ with finite element nodal vectors will be needed.

4.1.2.1 Practical Computation of the Expansion

It is now discussed how the parameters in the expansion may be computed. For non-Gaussian random fields $\kappa(x,\theta)$ the probability distributions of the uncorrelated random variables $\xi_1, \xi_2, \ldots$ in Eq. (4.17) are not analytically known. According to Eq. (3.4), the ξ_i are

$$\xi_i(\theta) = \frac{1}{\sqrt{\lambda_i}} \int_R \kappa(x,\theta)\kappa_i(x) \, dx. \tag{4.19}$$

If the polynomial chaos-expansion of $\kappa(x,\theta) = \sum_\iota \kappa^{(\iota)}(x) H_\iota(\theta)$ is known (see Section 4.1.1.2 on how to compute it), then the polynomial chaos expansion of the ξ_i may be computed numerically as

$$\xi_i(\theta) = \sum_{\iota \in (\mathbb{N}_0)\mathbb{N}_c} \xi_i^{(\iota)} H_\iota(\theta), \text{ where } \xi_i^{(\iota)} = \frac{1}{\sqrt{\lambda_i}} \int_R \kappa^{(\iota)}(x)\, \kappa_i(x)\, dx. \tag{4.20}$$

4.1.2.2 Using the Expansion in Practice

Inserting Eq. (4.18) into Eq. (3.47) yields the expansion

$$\boldsymbol{K}_{\alpha,\beta} = \sum_{i=0}^{\infty} \sum_{\iota \in (\mathbb{N}_0)_c^{\mathbb{N}}} \sqrt{\lambda_i}\, \xi_i^{(\iota)}\, \boldsymbol{K}_i\, \mathbf{E}\left(H_\alpha H_\beta H_\iota\right) \tag{4.21}$$

for the sub-matrices $\boldsymbol{K}_{\alpha,\beta}$ of the block matrix $\mathbf{K}$, where $\alpha, \beta \in \mathcal{I}$. According to Theorem 4.1, the sum over ι is unchanged if the infinite set $(\mathbb{N}_0)_c^{\mathbb{N}}$ is replaced by the finite set $\mathcal{J}(\mathcal{I})$ defined in Eq. (4.11). In the implementation the series is truncated after the l-th KL eigenmode and instead of the original block matrix its truncated version $\mathbf{K}^l$ is used. With the matrices $\Delta^{(\iota)}$ defined in Eq. (4.12), the approximation is

$$\mathbf{K} \approx \mathbf{K}^l := \sum_{i=0}^{l} \sum_{\iota \in \mathcal{J}(\mathcal{I})} \sqrt{\lambda_i} \xi_i^{(\iota)} \Delta^{(\iota)} \otimes \boldsymbol{K}_i. \tag{4.22}$$

Truncating the series in i after l terms is equivalent to replacing the original SPDE by an approximation, where $\kappa(x)$ is replaced by its l-term KL-expansion. The stability of such approximations was discussed in Section 3.3.1, where also the findings in [8] were mentioned according to which the approximated SPDE operator may become non-coercive. Whether this is the case may be checked numerically and is thus not considered here further.

As the operator $\mathbf{K}$ acts on a finite-dimensional space, stability can be more easily achieved than for the approximations of the SPDE discussed in Section 3.3.1. The KL-expansion converges in general only in $L^2(R) \otimes L^2(\Omega)$, but the following theorem [113] shows that even then the approximation Eq. (4.22) is stable:

Theorem 4.2: *A finite number of terms suffices to keep the operators $\boldsymbol{K}^l$ uniformly positive definite and thus their inverses uniformly bounded.*

PROOF: As linear operators on $\mathbb{R}^{n\cdot|\mathcal{I}|} \cong V^h \otimes (S)^{\mathcal{I}}$, the $\mathbf{K}, \mathbf{K}^0, \mathbf{K}^1, \ldots$ are elements of the finite-dimensional space $\mathbb{R}^{n\cdot|\mathcal{I}| \times n\cdot|\mathcal{I}|}$. All norms on a finite-dimensional space are equivalent. Thus, the convergence in L^2 implies uniform convergence and the truncation may be chosen such that the $\boldsymbol{K}^l$ are uniformly positive. □

It is now assumed that l is chosen such that $\mathbf{K}^l$ is positive definite. As was discussed in Section 3.1, the number of terms required for a good KL-approximation depends on how fast the covariance function cov_κ decays around the diagonal—the slower it decays, the less terms are needed.

Usually, $l \ll |\mathcal{I}|$ and the memory demands of this representation are hence significantly smaller than for the representation discussed in the previous section. Additionally, the execution of the block matrix-vector product is cheaper: it may be performed by computing $l \cdot |\mathcal{I}|$ matrix-vector products $\boldsymbol{K}_i \boldsymbol{u}_\beta$ and this number is usually a much smaller number than for the other representations. The following example illustrates this.

Example 4.2: As in Example 4.1, assume that the solution is expanded in 10-dimensional polynomial chaos of degree 4 and thus again $|\mathcal{I}| = |\mathcal{I}_{10,4}| = 1001$. Assume that $l = 50$ terms are kept in the KL-expansion of the operator.

To store $\mathbf{K}$ requires to store the $l \cdot |\mathcal{J}(\mathcal{I})| = |\mathcal{I}_{10,8}| \approx 2$ million coefficients $\xi_i^{(\iota)}$ and the fifty matrices $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_l$. Compared to the $43,785$ matrices required to be stored before the memory requirements are much smaller. The block matrix-vector product may be computed by $l \cdot |\mathcal{I}| \approx 50,000$ matrix-vector products, which requires significantly less effort than the half million matrix vector products required at least in the previous section.

Note that this representation is well-suited for a parallelisation. For example, Eq. (4.22) is a sum (in i) of linear operators that may be executed in parallel. This and other parallelisation aspects are discussed in Section 4.3.

4.2 Solvers for linear SPDEs

The block system of equations $\mathbf{Ku} = \mathbf{f}$ will be solved by iterative methods which require only the matrix-vector product with the block matrix $\mathbf{K}$. The block matrix $\mathbf{K}$ is never constructed explicitly.

4.2.1 Block Iterative Solvers

The Kronecker product representations of the previous sections are well-suited for an iterative solution of the linear block system: they allow to perform the vector product $\mathbf{Ku}$ efficiently and they allow to store the representation of $\mathbf{K}$ efficiently.

The tensor product structure of the equations suggests to use block versions of the iterative solvers where the sub-vectors $\boldsymbol{u}^{(\beta)}$ of the block vector $\mathbf{u}$ are treated as smallest units. Here, block versions of the classical iterative procedures are used, e.g. variants of block Jacobi- and SSOR-methods and Krylov space techniques. Their use and implementation has been described in [52, 135] and in [109–111].

Preconditioned block conjugate gradient solvers are also used here to solve linear elliptic SPDEs. These require preconditioners for a good convergence and variants of the above mentioned block versions of the classical iterative solvers are employed for this.

Let $\boldsymbol{M}_0$ be an approximate inverse of $\boldsymbol{K}_0 = \mathbf{E}(\boldsymbol{K}(\omega))$. For example, $\boldsymbol{M}_0$ may represent the execution of a few iterations of an iterative solver for the deterministic analogon of the SPDE.

This approximate inverse is used to implement an inexact block Jacobi-preconditioner that may be written as the block diagonal matrix

$$\mathbf{P} = \boldsymbol{I} \otimes \boldsymbol{M}_0 = \begin{pmatrix} \boldsymbol{M}_0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \boldsymbol{M}_0 \end{pmatrix}. \tag{4.23}$$

The inexact block JOR and block SOR solvers may be implemented similarly.

The preconditioning involves the solution of linear equations with the matrix $\mathbf{P}$ and this requires to solve $|\mathcal{I}|$ uncoupled systems of equations involving the matrix $\boldsymbol{M}_0 \in \mathbb{R}^{n\times n}$. Each of these linear systems has the size of the spatial discretisation and may be performed by calling the deterministic solver with the mean of κ as material. The numerical experiments in Section 4.2.3 will show that an approximate inverse $\boldsymbol{M}_0$ of the mean system matrix may be used in practice.

4.2.2 Multilevel Solution

The polynomial chaos is a hierarchical basis and it provides a natural multilevel structure to the equations. Solvers exploiting this structure are discussed now.

In particular, the multilevel method [115, 168] is implemented as follows: let $\mathbf{K}_{cc}$ be the block system matrix when using a "coarse" polynomial chaos ansatz. As the system matrix has a hierarchical structure, adding some additional "fine" ansatz functions yields the system matrix

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cf} \\ \mathbf{K}_{fc} & \mathbf{K}_{ff} \end{pmatrix}.$$

In the following, block iterative methods will be used as smoothers, e.g. block JOR or block SSOR smoothers. The effect of one iteration of the smoother on the matrix $\mathbf{K}$ will be denoted by $\mathbf{S}$. For example, one may choose $\mathbf{S} = \boldsymbol{I} \otimes \boldsymbol{M}_0$, just as the preconditioner in Eq. (4.23).

A two level scheme may then be constructed like this, where $\mathbf{u}_f$ is the current approximation:

$$\begin{array}{lcll} \mathbf{u}_f & := & \mathbf{S}^{\nu_1}\mathbf{u}_f & \text{(apply smoother } \nu_1 \in \mathbb{N} \text{ times)}, \\ \mathbf{r}_f & := & \mathbf{f} - \mathbf{K}\mathbf{u}_f & \text{(compute residual)}, \\ \mathbf{r}_c & := & \mathbf{I}_f^c \mathbf{r}_f & \text{(project residual into coarse space)}, \\ \Delta\mathbf{u}_c & := & \mathbf{K}_{cc}^{-1}\mathbf{r}_c & \text{(solve coarse problem to obtain coarse correction)}, \\ \mathbf{u}_f & := & \mathbf{u}_f + \mathbf{I}_c^f \Delta\mathbf{u}_c & \text{(prolongate coarse correction into fine level)}. \end{array}$$

Thus, the next iterate $\mathbf{u}_f^{(j+1)}$ is computed from the previous one $\mathbf{u}_f^{(j)}$ as

$$\mathbf{u}_f^{(j+1)} := \boldsymbol{u}_f^{(j)} + \mathbf{I}_c^f \mathbf{K}_{cc}^{-1} \mathbf{I}_f^c \left(\mathbf{f} - \mathbf{K}\mathbf{S}^{\nu_1}\mathbf{u}_f^{(j)}\right).$$

This scheme may recursively be extended into a multilevel scheme by replacing $\mathbf{K}_{cc}^{-1}$ by a similar two-level scheme. Apart from the smoothers that are chosen here as block iterative methods, the following ingredients are needed:

Restriction and Prolongation: The prolongation $\mathbf{I}_c^f$ defines the transfer from a coarse to a fine level. The restriction $\mathbf{I}_f^c$ is chosen as its transpose. A natural choice of $\mathbf{I}_c^f$ would be a sparse approximation of the Schur complement

$$\mathbf{I}_c^f = \begin{pmatrix} \mathbf{I} \\ -\mathbf{K}_{ff}^{-1}\mathbf{K}_{fc} \end{pmatrix},$$

but then the computation of the matrix $\mathbf{I}_c^f \mathbf{K}_{cc}^{-1} \mathbf{I}_f^c$ would be expensive and the advantages of the hierarchical structure would be lost. Tests with this prolongation showed poor overall performance in the multilevel scheme: the number of iterations was reduced, but the overall work was increased.

Because of the hierarchical structure, $\mathbf{K}_{cc}$ *is* the system matrix on the coarser level. Thus, $\mathbf{I}_c^f$ is chosen here as injection:

$$\mathbf{I}_c^f = \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix}.$$

Level Structure: The level structure is a sequence of nested subspaces of the ansatz space $V^h \otimes (S)^{\mathcal{I}}$. To allow using the deterministic code for the spatial discretisation and for the smoother, the levels are constructed here only in the stochastic dimensions.

Levels are chosen here as spaces $V^h \otimes (S)^{\mathcal{I}_1}, V^h \otimes (S)^{\mathcal{I}_2}, \ldots V^h \otimes (S)^{\mathcal{I}_L}$, where each $(S)^{\mathcal{I}_l}$ is the subspace of (S) with the basis $\{H_\alpha, \alpha \in \mathcal{I}_l\}$. Here, $\mathcal{I}_1 := \mathcal{I}$ denotes the finest and $\mathcal{I}_L$ the coarsest level.

As a *p*-method in the stochastic dimensions is used, it is natural to choose a level as the polynomial chaos of a given degree in a subset of the stochastic dimensions. Here, the coarsest level is chosen as $\mathcal{I}_1 := \{0\}$. As $V^h \otimes (S)^{\mathcal{I}_L} = V^h \otimes \{H_0\} \cong V^h$, the coarsest level corresponds to the deterministic analogon of the SPDE. The prolongation and restriction are defined as above as injection and orthogonal projection.

Hence, a multilevel algorithm is used where the levels are defined by the degrees and dimensions of the polynomial chaos and where the prolongation is chosen as injection.

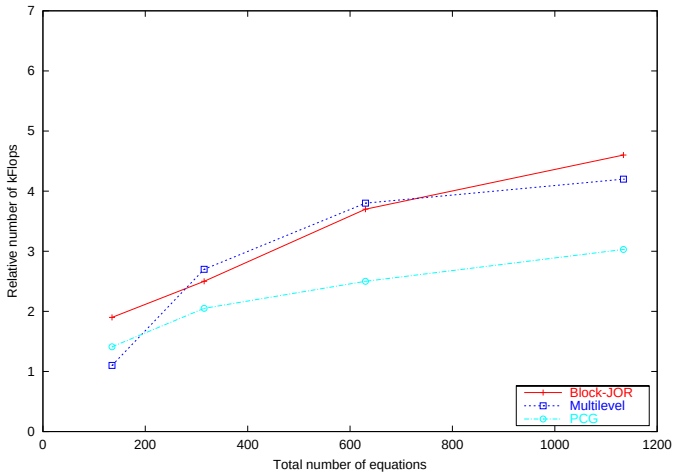
4.2.3 Numerical Tests

Various numerical experiments were performed for varying parameters of the SPDE and of the discretisation. If not stated otherwise, then the parameters were the following in each experiment: The KL-eigenproblem was discretised on an 8×8 regular grid using six Gaussian mutually independent basic random variables. The operator was expanded in polynomial chaos of order 4. The ansatz space was constructed by discretising the unit rectangle by a 4×4 grid and by polynomial chaos spaces of degrees up to 4.

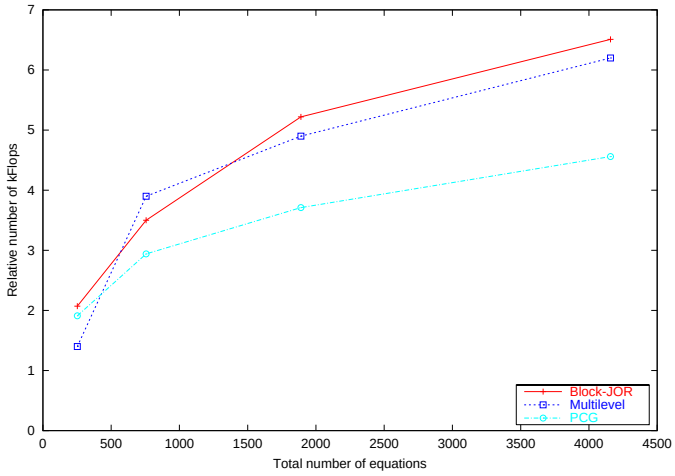
The solvers used were block JOR, preconditioned conjugate gradients (PCG), and the *V*-cycle multilevel solver. The preconditioner for the conjugate gradients solver was $\boldsymbol{I} \otimes \boldsymbol{M}_0$, where the matrix $\boldsymbol{M}_0$ was obtained from the deterministic solver using the mean μ_κ as material. The smoother for the multilevel solver was block JOR. As the goal is here to investigate the growth of the effort per equation and hence with the number of unknowns, the ordinates show the number of flops divided by the number of equations—here denoted as "relative flops"—and the abscissae show the total number of equations.

In Fig. 4.2 the degree of the polynomial chaos is varied from 1 to 4. The multilevel scheme performs a bit better than the block JOR scheme, while the PCG scheme shows the best performance.

The number of stochastic dimensions was varied for Fig. 4.3 while keeping the degree of the polynomial chaos constant. Here, the multilevel scheme converged worse than PCG and block JOR.



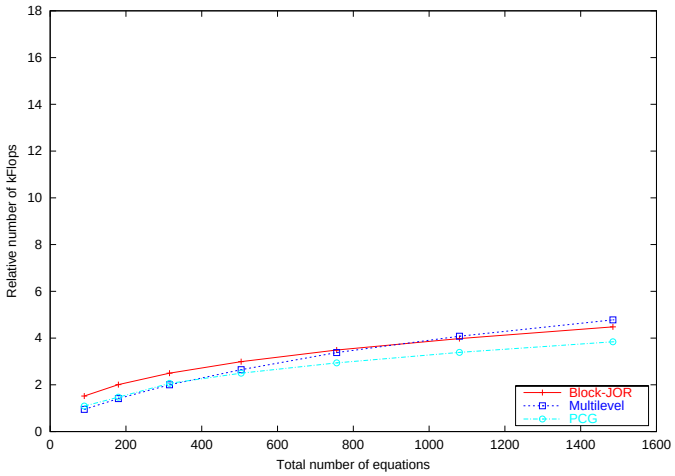
(a) Four-dimensional polynomial chaos



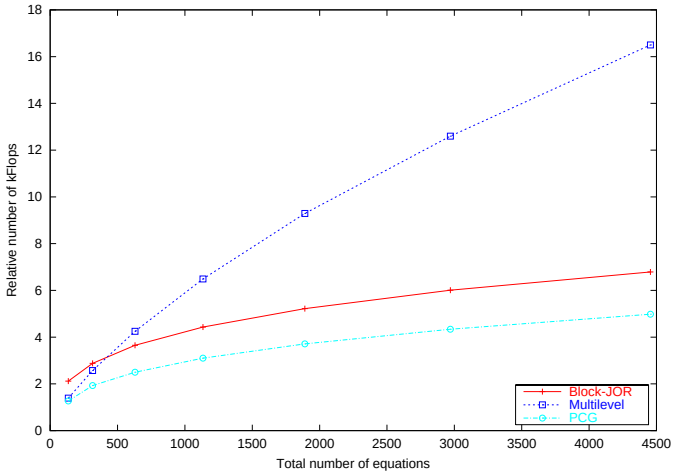
(b) Six-dimensional polynomial chaos

Figure 4.2: Degree of polynomial chaos varied from 1 to 4.

The stochastic discretisation was kept constant in Fig. 4.4, but the coefficient of variance of the stochastic heat conductivity was varied. This plot illustrates that the numerical costs increase with the coefficient of variance of the underlying fields and that a higher order approximation of the stochastic field leads to higher costs. The reason may be that with a higher coefficient of variation the SPDE varies stronger about its mean, which diminishes the quality of the mean as



(a) Polynomial chaos of degree 3



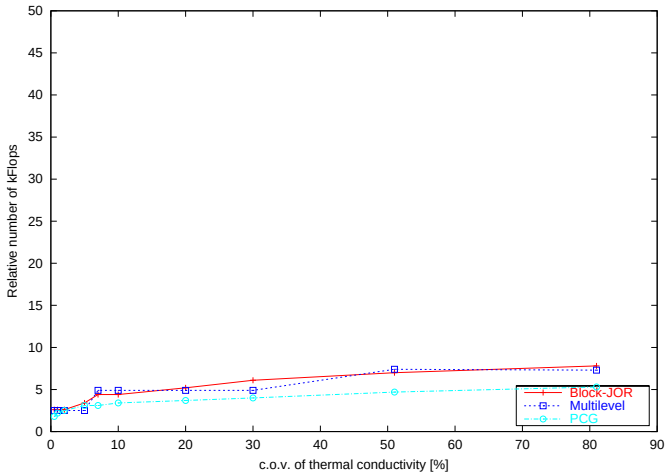
(b) Polynomial chaos of degree 4

Figure 4.3: Stochastic dimension varied from 2 to 8.

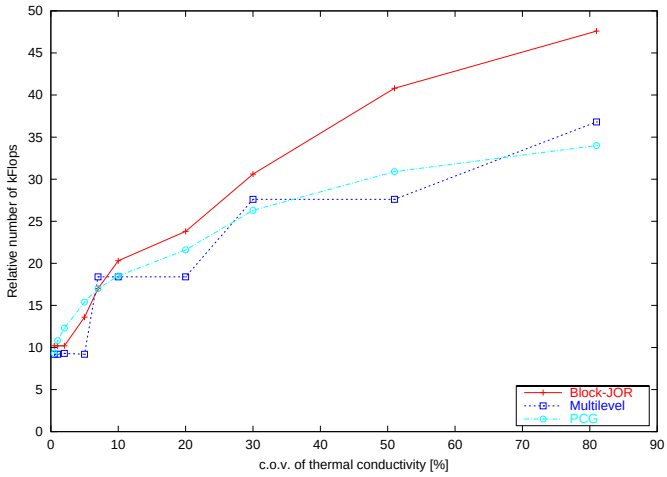
preconditioner.

Fig. 4.5 demonstrates that increasing the number of spatial ansatz function has only a small effect on the relative effort of the preconditioned conjugate gradients scheme. As the inverse of the mean matrix was used as preconditioner, this was to be expected.

In all plots, the relative effort for solving the system grows slowly with the



(a) Thermal conductivity expanded in polynomial chaos of degree 2



(b) Thermal conductivity expanded in polynomial chaos of degree 4

Figure 4.4: Coeffi cient of variance is varied from 0 to 0.9

number of unknowns. It may be concluded from the behaviour of the block JOR solver that the matrices stay well-conditioned, when the stochastic ansatz space is enlarged. In all plots, the preconditioned conjugate gradients solver performs best.

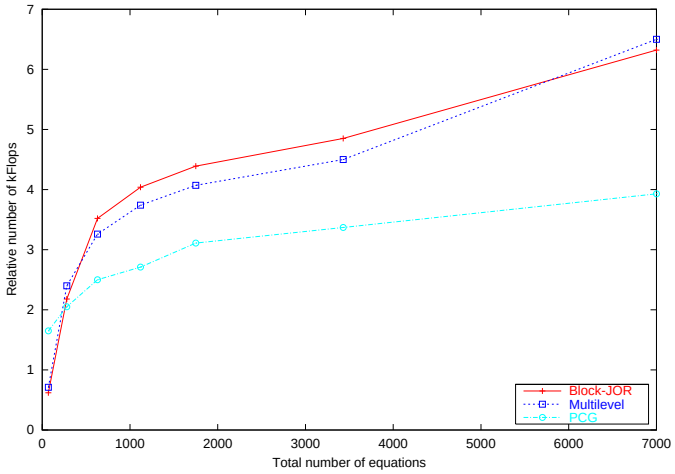


Figure 4.5: Spatial discretisation varied from 1 to 100.

4.2.4 Conclusions

The classical iterative and semi-iterative are well-suited to solve linear elliptic SPDEs efficiently. These methods work well as the main contribution on the diagonal blocks of the block equations is the deterministic matrix $\boldsymbol{K}_0$.

The multilevel schemes also show good converge-behaviour, but they do not show the hoped for behaviour, i.e. costs growing linearly in the number of unknowns. While the plots shown here were not obtained using a full multigrid-algorithm, the same qualitative behaviour was observed for full multigrid.

The multilevel scheme for stochastic PDEs may be obtained by discretising the stochastic dimensions of the stochastic PDE before the spatial dimensions. This yields a partial differential equation for every stochastic ansatz function and all these partial differential equations are coupled by the stochastic discretisation. Hence, although it is not obvious, discretised systems of many coupled partial differential equations are solved and implementing efficient multigrid algorithms for systems of PDEs is still an active area of research.

As the convergence speed of the iterative solvers is not affected strongly by enlarging the ansatz space it may be concluded that the condition number of the system matrices grows slowly. This hints at the polynomial chaos being a good basis for discretising the type of stochastic PDEs considered here.

It was shown that the solution may be sped up by reusing an existing solver for the deterministic problem as preconditioner. The black-box integration of an existing software is possible and simplifies the development of SPDE software considerably; see Chapter 5 for a discussion of software aspects.

4.3 A Parallel Solver for Linear SPDEs

The simulation of realistic problems [158] requires large ansatz-spaces. Hence, parallel techniques are needed. This section discusses a parallel solver for the system of linear block equations Eq. (4.1), earlier versions of which are described in [83, 85, 86]. The parallel solver is based on a conjugate gradients solver preconditioned by a block Jacobi method (see Section 4.2.1). The parallelisation of the solver is performed by parallelising the block matrix-vector product $\mathbf{v} := \mathbf{K}\mathbf{u}$ and by parallelising the preconditioner.

For the parallelisation of the block matrix-vector product, the tensor product structure of the representation discussed in Section 4.1.2 is exploited. The parallelisation is performed in a hierarchical manner and allows to exploit different kinds of parallelism to adapt the parallelisation to the problem at hand.

4.3.1 Parallel Block Matrix-Vector Product: Data Distribution

First, the parallelisation of the block matrix-vector product $\mathbf{v} := \mathbf{K}\mathbf{u} \approx \mathbf{K}^l\mathbf{u}$ is discussed, where $\mathbf{K}^l$ is given by Eq. (4.22). The sub-vectors of the result are

$$\boldsymbol{v}^{(\alpha)} = (\mathbf{K}^l\mathbf{u})_\alpha = \sum_{i=0}^{l} \sum_{\beta\in\mathcal{I}} \sum_{\iota\in\mathcal{J}(\mathcal{I})} \sqrt{\lambda_i}\xi_i^{(\iota)}\Delta_{\alpha,\beta}^{(\iota)}\boldsymbol{K}_i\,\boldsymbol{u}^{(\beta)}, \qquad \alpha\in\mathcal{I}. \tag{4.24}$$

The parallelisation is performed on different levels:

Parallel deterministic code: Remember that the discretisation of the spatial domain is performed by some existing software that is called the *deterministic code*. Each matrix $\boldsymbol{K}_i$ may be identified with an instance of this deterministic code, with the material set to $\kappa_i(x)$, the i-th KL-eigenfunction of $\kappa(x,\omega)$.

The deterministic code may be a parallel program, for example based on a domain decomposition of the spatial domain. A set of processors running one instance of the deterministic code will be called a "processor group" (p-group) and the available processors will be divided into N_P "processor groups" labelled $pg_1,\ldots,pg_{N_P}$. Every p-group stores one or more of the matrices $\boldsymbol{K}_i$ and a set of sub-vectors of the block vectors $\mathbf{u}$ and $\mathbf{v}$.

Every matrix $\boldsymbol{K}_i$ is distributed over the processors in the p-group as required by the deterministic code. The sub-vectors $\boldsymbol{u}^{(\beta)}$ and $\boldsymbol{v}^{(\alpha)}$ are coefficient vectors for the spatial discretisation and their arrangement is left to the deterministic code.

The processor groups are the smallest building blocks of the parallel algorithm upon which coarser levels of parallelism are built. This results in a hierarchical parallel execution. What matrices $\boldsymbol{K}_i$ and what sub-vectors $\mathbf{u}$ and $\mathbf{v}$ are stored on what processor group depends on the coarser levels of the parallelisation.

Simultaneous execution of deterministic codes: According to Eq. (4.22), the block matrix $\mathbf{K}^l$ is a sum (over i) of tensor products. Apparently, the block matrix-vector products in this sum may be computed in parallel by parallelising the sum over i. For this, the matrices $\boldsymbol{K}_i$ are distributed as evenly as possible over the processor groups.

It may be favourable for efficiency reasons to run identical instances of the $\boldsymbol{K}_i$ simultaneously on more than on one processor group. The matrices are then replicated N_K times and redundant copies of each matrix are stored on N_k different p-groups. In the following, the number of matrix replications N_K is used to characterise how the matrices are distributed over the p-groups.

The matrices $\Delta^{(\iota)}$ are stored as a sorted list of non-zero entries on every processor. If a matrix $\boldsymbol{K}_i$ is stored on a processor, then this processor holds a copy of all $\xi_i^{(\iota)}, \iota \in \mathcal{J}(\mathcal{I})$. Every set of processor groups that holds all $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_l$ can therefore apply the whole block matrix-vector product to the part of the block vector $\mathbf{u}$ stored there and can add it to the appropriate parts of the right hand side. Such a set of processor groups will be called an *operator group*.

As more than one $\boldsymbol{K}_i$ may be stored on a processor group, it may be necessary to exchange the material parameter in the deterministic code, just as in the sequential case. For this, the deterministic code may either be requested to set the material by calling an appropriate function, or it may be restarted with another material parameter.

Distributed block vectors: To allow large ansatz spaces, the sub-vectors of $\mathbf{u}$ and $\mathbf{v}$ are distributed as evenly as possible over the processor groups, thus parallelising both the sum in β and the loop over α in Eq. (4.24).

It will be seen that it may be advantageous to store every block vector more than once (to replicate it). This increases the memory requirements but reduces the execution time. Hence, copies of each block vector are stored on N_V different sets of p-groups. In the following, the number of block vector replications N_V characterises their distribution.

Before and after the block matrix-vector product, all block vectors are arranged in the same manner. During the block matrix-vector multiplication, parts of them need to be exchanged between the processor groups; this is discussed in the next section.

4.3.2 Hierarchical Parallel Matrix-Vector Product

It is now discussed what communication is necessary and how much work is required on each processor if the Algorithm 4.1 for the block matrix-vector product Eq. (4.24) is parallelised.

1: **for all** $\alpha \in \mathcal{I}$ **do**
2: $\quad \boldsymbol{v}^{(\alpha)} \leftarrow 0$
3: **end for**
4: **for all** $i = 0, \ldots, l$ **do**
5: $\quad$ Activate deterministic solver $\boldsymbol{K}_i$ (set material)
6: $\quad$ **for all** $\beta \in \mathcal{I}$ **do**
7: $\quad\quad \boldsymbol{w}_{i,\beta} := \mathbf{K}_i \boldsymbol{u}^{(\beta)}$
8: $\quad\quad$ **for all** $\alpha \in \mathcal{I}$ **do**
9: $\quad\quad\quad c_{i,\beta,\alpha} := \sum_{\iota \in \mathcal{J}(\mathcal{I})} \sqrt{\lambda_i} \xi_i^{(\iota)} \Delta_{\alpha,\beta}^{(\iota)}$
10: $\quad\quad\quad \boldsymbol{v}^{(\alpha)} \leftarrow c_{i,\beta,\alpha} \boldsymbol{w}_{i,\beta} + \boldsymbol{v}^{(\alpha)}$
11: $\quad\quad$ **end for**
12: $\quad$ **end for**
13: **end for**

Algorithm 4.1: Sequential Block Matrix-Vector Product

Every product $\boldsymbol{K}_i \boldsymbol{u}^{(\beta)}$ contributes to every $\boldsymbol{v}^{(\alpha)}$. Hence, data must be exchanged between the processors if the matrices $\boldsymbol{K}_i$ or the block vectors **u** and **v** are stored in a distributed manner. Most communication may be performed as cyclical shifts of the block vectors **u** and **v** across subsets of the processor groups (the first p-group in the set sends its part of the block vector to the second, the second sends to the third, and so on. The last p-group sends its part to the first p-group):

p-group	block vector	shift	after Shift
pg_1	$\boldsymbol{u}^{(1)}$	$\searrow$	$\boldsymbol{u}^{(N_{pg})}$
pg_2	$\boldsymbol{u}^{(2)}$	$\searrow$	$\boldsymbol{u}^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$pg_{N_{pg}}$	$\boldsymbol{u}^{(N_{pg})}$	$\searrow$	$\boldsymbol{u}^{(N_{pg}-1)}$

Table 4.2: Cyclical Shift over all Processor Groups

The algorithm for the parallel multiplication will be discussed in Section 4.3.3. Before this, the communication requirements are visualised by examples. For simplicity, all examples use four processor groups, four matrices $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_3$, and block vectors **u** and **v** with 12 sub-vectors.

4.3.2.1 No Redundancy

For the most efficient distribution of the data in terms of memory demands, every matrix and every vector is stored only once.

p-group	$\boldsymbol{K}_i$	**u**	**v**
pg_1	$\boldsymbol{K}_0$	$\boldsymbol{u}^{(1)} \cdots \boldsymbol{u}^{(3)}$	$\boldsymbol{v}^{(1)} \cdots \boldsymbol{v}^{(3)}$
pg_2	$\boldsymbol{K}_1$	$\boldsymbol{u}^{(4)} \cdots \boldsymbol{u}^{(6)}$	$\boldsymbol{v}^{(4)} \cdots \boldsymbol{v}^{(6)}$
pg_3	$\boldsymbol{K}_2$	$\boldsymbol{u}^{(7)} \cdots \boldsymbol{u}^{(9)}$	$\boldsymbol{v}^{(7)} \cdots \boldsymbol{v}^{(9)}$
pg_4	$\boldsymbol{K}_3$	$\boldsymbol{u}^{(10)} \cdots \boldsymbol{u}^{(12)}$	$\boldsymbol{v}^{(10)} \cdots \boldsymbol{v}^{(12)}$

Table 4.3: Memory layout for parallel matrix-vector-product, most effi cient memory usage

Example 4.3: Table 4.3 illustrates this case. As every combination $\boldsymbol{K}_i\boldsymbol{u}^{(\beta)}$ contributes to every $\boldsymbol{v}^{(\alpha)}$, four cyclical shifts of **u** across all p-groups are necessary. For every configuration of **u** it is necessary to shift **v** four times cyclically across all p-groups. In total, 16 cyclical shifts of **v** are needed.

In general, if there is no redundancy as in the above example, then N_{pg} cyclical shifts of **u** and N_{pg}^2 cyclical shifts of **v** across all processor groups are required.

According to Algorithm 4.1, every processor group must perform the following work for every configuration of **u** and **v**: The vectors $\boldsymbol{w}_{i,\beta} := \boldsymbol{K}_i\boldsymbol{u}^{(\beta)}$ must be computed for all local $\boldsymbol{K}_i$ and for all local $\boldsymbol{u}^{(\beta)}$ and $c_{i,\beta,\alpha}\boldsymbol{w}_{i,\beta}$ must be added to each local $\boldsymbol{v}^{(\alpha)}$ (this is called an axpy operation, see Algorithm 4.1, line 10).

Note that it is assumed here that no auxiliary block vector is used that stores the products $\boldsymbol{w}_{i,\beta} := \boldsymbol{K}_i\boldsymbol{u}^{(\beta)}$. In contrast to the sequential Algorithm 4.1, the vector $\boldsymbol{K}_i\boldsymbol{u}^{(\beta)}$ is added only to the sub-vectors of **v** that are locally available. Hence, after shifting the block vector **v**, the products $\boldsymbol{w}_{i,\beta} := \boldsymbol{K}_i\boldsymbol{u}^{(\beta)}$ must be computed again to obtain the contributions to the parts of **v** that were stored on other processors before the shift was performed.

Every processor group stores $(l+1)/N_{pg}$ of the matrices $\boldsymbol{K}_i$ and $|\mathcal{I}|/N_{pg}$ subvectors of **u** and **v** (roundoff errors are ignored for simplicity). Hence, $(l+1)/N_{pg} \cdot |\mathcal{I}|/N_{pg}$ matrix vector multiplications and $(l+1)/N_{pg} \cdot (|\mathcal{I}|/N_{pg})^2$ axpy operations are executed for every configuration of **u** and **v**. In total, there are N_{pg}^2 such configurations. Hence, the execution time of the block-matrix vector product on each processor is

$$T_{bmv,1} = (l+1)|\mathcal{I}|\,T_{mv} + (l+1)|\mathcal{I}|^2\,T_{axpy} + (N_{pg}+N_{pg}^2)\,T_{shift}, \tag{4.25}$$

where T_{mv} denotes the time required to perform one matrix-vector product $\boldsymbol{K}_i\boldsymbol{u}^{(\beta)}$, where T_{axpy} is the duration of one axpy operation $\boldsymbol{v}^{(\alpha)} \leftarrow \boldsymbol{v}^{(\alpha)} + c_{i,\beta,\alpha}\boldsymbol{w}_{i,\beta}$, and where T_{shift} is the duration of a cyclical shift of a block vector across all processor groups.

According to Eq. (4.25) increasing the number of processors does not reduce the execution time, but increases it by the communication overhead. It may sur-

p-group	$\boldsymbol{K}_i$	**u**	**v**
pg_1	$\boldsymbol{K}_0\ \boldsymbol{K}_1$	$\boldsymbol{u}^{(1)} \cdots \boldsymbol{u}^{(3)}$	$\boldsymbol{v}^{(1)} \cdots \boldsymbol{v}^{(3)}$
pg_2	$\boldsymbol{K}_2\ \boldsymbol{K}_3$	$\boldsymbol{u}^{(4)} \cdots \boldsymbol{u}^{(6)}$	$\boldsymbol{v}^{(4)} \cdots \boldsymbol{v}^{(6)}$
pg_3	$\boldsymbol{K}_0\ \boldsymbol{K}_1$	$\boldsymbol{u}^{(7)} \cdots \boldsymbol{u}^{(9)}$	$\boldsymbol{v}^{(7)} \cdots \boldsymbol{v}^{(9)}$
pg_4	$\boldsymbol{K}_2\ \boldsymbol{K}_3$	$\boldsymbol{u}^{(10)} \cdots \boldsymbol{u}^{(12)}$	$\boldsymbol{v}^{(10)} \cdots \boldsymbol{v}^{(12)}$

Table 4.4: Memory layout for parallel matrix-vector-product, replicated operator

prise that no speedup is obtained. However, parallelisations of matrix-vector products usually store the matrix in full form and then a good speedup may be obtained by distributing the rows of the matrix over the processors. Here the matrix is stored in the Kronecker product form Eq. (4.22). As every possible combination $\boldsymbol{K}_i\boldsymbol{u}^{(\beta)}$ is required to compute the effect of any sub-matrix of **K**, no speedup is obtained here.

This type of parallelisation may still be advantageous as the available memory for the block vectors **u** and **v** scales linearly with the numbers of processors. Hence, this form of parallelisation allows large ansatz spaces. The block diagonal preconditioner will be parallelised with almost perfect speedup. Hence, acceptable efficiencies are obtained in toto (see Section 4.3.6).

4.3.2.2 Replication of the Operator

To increase the parallel efficiency, redundancies are introduced. First, the replication of the matrices $\boldsymbol{K}_i$ is considered. This type of redundancy is needed anyway as the number of p-groups may be larger than the number of matrices $\boldsymbol{K}_i$.

Example 4.4: Table 4.4 shows the memory layout if two copies of each $\boldsymbol{K}_i$ are stored and if the block vectors are stored without redundancy. Again, **u** must be shifted cyclically four times across all processor groups. For every configuration of **u**, only two cyclical shifts of **v** inside the p-group set $\{pg_1, pg_2\}$ and inside the p-group set $\{pg_3, pg_4\}$ are required as each of these p-group sets holds a complete operator.

Let N_K be a divisor of N_{pg} and divide the processor groups into N_K sets that will be called *operator groups*. Every operator group comprises $S_{og} := N_{pg}/N_K$ p-groups and holds one copy of every $\boldsymbol{K}_i$, $i = 0, \ldots, l$. For example, Table 4.4 contains the two operator groups $\{pg_1, pg_2\}$ and $\{pg_3, pg_4\}$, each having the size $S_{og} = 2$. Let the matrices be distributed as evenly as possible across the p-groups.

Then every p-group stores at most $\lceil (l+1)/S_{og} \rceil$ of the matrices. If the total number of matrices $l+1$ is not a multiple of S_{og}, then some p-groups hold one matrix more than the others, resulting in a slightly unevenly distributed work load.

The parallel block matrix-vector product requires N_{pg} shifts of **u** across all processor groups. For every configuration of **u** it is necessary to perform S_{og} cyclical shifts of **v** inside every operator group. The shifts inside the operator groups are independent of each other. Hence, the effort of shifting **v** once inside every operator group is comparable to the effort of shifting **v** once across all p-groups. In total this amounts to $N_{pg} S_{og} = N_{pg}^2 / N_K$ cyclical shifts of **v**.

The following work must be performed on every processor group for every configuration of **v** and **u**: Every $c_{i,\beta,\alpha} \cdot \boldsymbol{K}_i \boldsymbol{u}^{(\beta)}$ (see Algorithm 4.1, line 10) must be added to every local $\boldsymbol{v}^{(\alpha)}$ for every local $\boldsymbol{K}_i$ and for every local $\boldsymbol{u}^{(\beta)}$. Hence, $(l+1)/S_{og} \cdot |\mathcal{I}|/N_{pg}$ matrix-vector products and $(l+1)/S_{og} \cdot (|\mathcal{I}|/N_{pg})^2$ axpy operations must be performed for every configuration of **v** and **u**.

As N_{pg} cyclical shifts of **u** and S_{og} cyclical shifts of **v** are required for every configuration of **u**, the total work required on each processor is

$$T_{bmv,2} = (l+1)|\mathcal{I}|\, T_{mv} + (l+1)|\mathcal{I}|^2\, T_{axpy} + N_{pg}\, S_{og}\, T_{shift}, \tag{4.26}$$

where the same symbols as in Eq. (4.25) are used.

The communication requirements are smaller than for the redundancy-free case. But again, no speedup results. Nonetheless, as for the redundancy-free case, this type of parallelisation may still be advantageous as it allows large ansatz spaces and as the combination with the parallelised block diagonal preconditioner yields acceptable parallel efficiencies.

4.3.2.3 Replication of the Block Vectors

For efficiency reasons it may be advantageous to hold more than one copy of every block vector. For example, the exchange of the material parameter may require a restart of the deterministic solver. In this case it may be advantageous to assign only one material parameter to each p-group, i.e. to run each deterministic solver instance with a fixed material parameter.

Example 4.5: Table 4.5 demonstrates this. The required communication may be performed in the following manner: **u** is cycled twice inside both p-group sets $\{pg_1, pg_2\}$ and $\{pg_3, pg_4\}$. For every configuration of **u**, the block vector **v** is shifted twice within the sets $\{pg_1, pg_2\}$ and $\{pg_3, pg_4\}$. After this, the $\boldsymbol{v}^{(\alpha)}$ on $\{pg_1, pg_2\}$ are added to their counterparts on $\{pg_3, pg_4\}$ by a parallel prefix operation. Finally, the results are redistributed over the processors.

Let N_{vec} be a divisor of N_{pg} and let the processors be arranged in N_{vec} sets of processor groups that will be called *vector groups*. Every vector group comprises

p-group	$\boldsymbol{K}_i$	**u**	**v**
pg_1	$\boldsymbol{K}_0$	$\boldsymbol{u}^{(1)} \cdots \boldsymbol{u}^{(6)}$	$\boldsymbol{v}^{(1)} \cdots \boldsymbol{v}^{(6)}$
pg_2	$\boldsymbol{K}_1$	$\boldsymbol{u}^{(7)} \cdots \boldsymbol{u}^{(12)}$	$\boldsymbol{v}^{(7)} \cdots \boldsymbol{v}^{(12)}$
pg_3	$\boldsymbol{K}_2$	$\boldsymbol{u}^{(1)} \cdots \boldsymbol{u}^{(6)}$	$\boldsymbol{v}^{(1)} \cdots \boldsymbol{v}^{(6)}$
pg_4	$\boldsymbol{K}_3$	$\boldsymbol{u}^{(7)} \cdots \boldsymbol{u}^{(12)}$	$\boldsymbol{v}^{(7)} \cdots \boldsymbol{v}^{(12)}$

Table 4.5: Memory layout for parallel matrix-vector-product without redundancy in operator and with two copies of block vectors.

$S_{vg} := N_{pg}/N_{vec}$ processor groups and holds one copy of **u** and of **v**. For example, Table 4.5 contains the two vector groups $\{pg_1, pg_2\}$ and $\{pg_3, pg_4\}$ of size $S_{vg} = 2$. Let the sub-vectors of the block vectors be distributed as evenly as possible. Then every p-group holds at most $\lceil |\mathcal{I}|/S_{vg} \rceil$ sub-vectors of every block vector. If S_{vg} is not a divisor of $|\mathcal{I}|$, then some of the p-groups hold one sub-vector less than the others. This leads to a slightly unevenly distributed work load, but the effects are negligible if the number of sub-vectors stored on every p-group is large.

Assume that the operators are not replicated. The communication requirements in the parallel block matrix-vector product are then S_{vg} cyclical shifts of **u** inside every vector group. As the shifts inside the vector groups are independent of each other, the effort of shifting **u** once inside every vector group is comparable to the effort of shifting **u** once across all p-groups. For every configuration of **u** it is necessary to perform S_{vg} cyclical shifts of **v** inside every vector group. Thus, in total, $S_{vg} = N_{pg}/N_{vec}$ cyclical shifts of **u** and $S_{vg}^2 = (N_{pg}/N_{vec})^2$ cyclical shifts of **v** are executed.

These steps exchange sub-vectors only within each vector group. It is therefore necessary to add up the **v** from all vector groups afterwards and redistribute the result to all processors. This is implemented by a parallel prefix operation (in the communication library MPI [117] this is called an "Allreduce"-operation).

The effort required on every processor for every configuration of **u** and of $\boldsymbol{v}$ is the following: $(l+1)/N_{pg} \cdot |\mathcal{I}|/S_{vg}$ matrix-vector multiplications $\boldsymbol{K}_i \boldsymbol{u}^{(\beta)}$ and $(l+1)/N_{pg} \cdot (|\mathcal{I}|/S_{vg})^2$ axpy operations $\boldsymbol{v}^{(\alpha)} \leftarrow c_{i,\beta,\alpha} \boldsymbol{w}_{i,\beta} + \boldsymbol{v}^{(\alpha)}$ (see Algorithm 4.1, line 10) are required.

As S_{vg} and S_{vg}^2 cyclical shifts are required for **u** and **v**, the total effort on every processor group is

$$T_{bmv,3} = S_{vg} \frac{l+1}{N_{pg}} |\mathcal{I}| \, T_{mv} + \frac{l+1}{N_{pg}} |\mathcal{I}|^2 \, T_{axpy} + (S_{vg} + S_{vg}^2) \, T_{shift} + T_{addup}, \quad (4.27)$$

where the same symbols as in Eq. (4.25) are used. The symbol T_{addup} denotes the execution time of the parallel prefix operation.

As Eq. (4.27) indicates and as the experiments in Section 4.3.6 show, this parallelisation may yield good speedups.

If the block vectors are stored with highest redundancy, then every processor group stores the complete block vectors and the only required communication is a parallel prefix operation (T_{shift} is then equal to zero). Accordingly, the measurements for this parallelisation in Section 4.3.6 show almost perfect efficiency for this case. However, the available memory does not scale with the number of p-groups in this case as it requires the number of block vector replications to be equal to the number of processor groups. If the number of block vector replications is instead kept constant for a growing number of p-groups, then the memory scales linearly with the number of p-groups and good speedups are possible according to Eq. (4.27). The available memory may also be increased by enlarging the processor groups, i.e. by increasing the parallelism in the deterministic solver.

4.3.3 Parallel Matrix-Vector Product, Algorithm

Every processor runs the pseudocode shown in algorithm 4.2 to execute the parallel block matrix-vector product. The following notation is used:

N_{pg} is the number of processor groups, a divisor of the number of processors.

$local_{dof}$ denotes the part of a sub-vector $\boldsymbol{u}^{(\beta)}$ or $\boldsymbol{v}^{(\alpha)}$ stored locally on a processor. For example, $\boldsymbol{v}^{(\alpha)}(local\_dof)$ is the part of the vector $\boldsymbol{v}^{(\alpha)}$ stored on the processor executing the algorithm.

N_K is the number of operator replications. It is a divisor of N_{pg} and if $N_K > 1$ then $N_{vec} = 1$.

$local_K$ is a set of integers. It denotes the matrices $\boldsymbol{K}_i$ stored locally on the processor executing the algorithm.

N_{vec} is the number of block vector replications. It is a divisor of N_{pg} and if $N_{vec} > 1$ then $N_K = 1$.

S_{vg} is the number of p-groups in every vector group, $S_{vg} = N_{pg}/N_{vec}$.

$local_{\mathbf{u}}$ denotes the set of sub-vectors of the block vector $\mathbf{u}$ stored locally on the executing processor at the moment of execution. Analogously, $local_{\mathbf{v}}$ does this for $\mathbf{v}$.

4.3.4 Parallel Solver

To solve the linear block system, a preconditioned conjugate gradients algorithm with the parallel block matrix-vector product discussed above is used. The preconditioner is a parallel version of the block diagonal preconditioner discussed in Section 4.2.1.

1: **for all** $\alpha \in local_{\mathbf{v}}$ **do**
2: $\quad \boldsymbol{v}^{(\alpha)}(local_{dof}) \leftarrow 0$
3: **end for**
4: **for all** $i \in local_K$ **do**
5: $\quad$ Activate deterministic solver $\boldsymbol{K}_i$ (set material)
6: $\quad$ **for** S_{vg} times **do** {cycle **u**}
7: $\quad\quad$ **for all** $\beta \in local_{\mathbf{u}}$ **do**
8: $\quad\quad\quad$ **perform collective operation** in processor group
9: $\quad\quad\quad\quad \boldsymbol{w} := \mathbf{K}_i \boldsymbol{u}^{(\beta)}$
10: $\quad\quad\quad$ **end collective operation**
11: $\quad\quad\quad$ **for** S_{vg}/N_K times **do**
12: $\quad\quad\quad\quad$ **for all** $\alpha \in local_{\mathbf{v}}$ **do**
13: $\quad\quad\quad\quad\quad c_{i,\beta,\alpha} := \sum_{\iota \in \mathcal{J}(\mathcal{I})} \sqrt{\lambda_i}\xi_i^{(\iota)}\Delta_{\alpha,\beta}^{(\iota)}$
14: $\quad\quad\quad\quad\quad \boldsymbol{v}^{(\alpha)}(local_{dof}) \leftarrow c_{i,\beta,\alpha}\boldsymbol{w}(local_{dof}) + \boldsymbol{v}^{(\alpha)}(local_{dof})$
15: $\quad\quad\quad\quad$ **end for**
16: $\quad\quad\quad\quad$ **perform collective operation** in operator group
17: $\quad\quad\quad\quad\quad$ perform cyclical shift of **v** inside operator group
18: $\quad\quad\quad\quad\quad$ obtain new $local_{\mathbf{v}}$
19: $\quad\quad\quad\quad$ **end collective operation**
20: $\quad\quad\quad$ **end for**
21: $\quad\quad$ **end for**
22: $\quad\quad$ **perform collective operation** in intersection of vector & matrix group
23: $\quad\quad\quad$ perform cyclical shift of **u** in intersection of vector & matrix group
24: $\quad\quad\quad$ obtain new $local_{\mathbf{u}}$
25: $\quad\quad$ **end collective operation**
26: $\quad$ **end for**
27: **end for**
28: **perform collective operation** in all processors {Allreduce operation}
29: $\quad$ sum up all the $\boldsymbol{v}^{(\alpha)}$ from all vector groups
30: $\quad$ redistribute the sum to all vector groups
31: **end collective operation**

Algorithm 4.2: Parallel Block Matrix-Vector Product. Note that comments are set in braces.

The block diagonal preconditioner applies the (maybe parallel) deterministic solver to every component of a block vector. Its parallelisation is straightforward: The deterministic solver is applied to the local parts of the block vector on the respective processor group. If the block vectors are stored redundantly, then every component of the result is computed on only one processor group and the results are distributed over the processors, afterwards. This yields a good parallel

efficiency for the block preconditioning stage.

4.3.5 Implementation

The parallel solver was implemented in portable C++ [167] using the portable communication library MPI (Message Passing Interface) [64, 117]. It was tested on a Linux cluster and on a CRAY T3E at the John von Neumann Institute for Computing at the Jülich research center [42]. Care was taken to make the solver portable and it was compiled with the GNU compilers [41] and with the Cray system compiler CC [32].

The deterministic solver is currently parallel preconditioned conjugate gradients solver based on the parallel software libray PetSC [13]. The computation of the matrices $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_l$ and of the coefficients $\xi_i^{(\iota)}$ is at the moment performed by the Matlab [106] part of the StoFEL library. The matrices $\Delta_{\alpha,\beta}^{(\iota)}$ are computed in parallel by a subroutine of the C++ module.

The Matlab-part and the parallel solver may communicate interactively via TCP/IP or via files. After its start, the parallel solver either reads a batch-file or opens a socket. The Matlab-part may connect to the socket and send commands to the parallel solver and receive results from it. Alternatively, the Matlab-part may write a batch file for the parallel solver. The TCP/IP communication was implemented by extending Matlab with MEX-files [105]. As Matlab and the parallel computer may run on different computer architectures, the External Data Representation standard XDR [170] is used for the data exchange.

4.3.6 Experiments and Parallel Efficiency

Several experiments were performed to test the parallel solver. All experiments were performed on a CRAY T3E at the John von Neumann Institute for Computing at the Jülich research centre [42]. This support by the Jülich research centre is gratefully acknowledged.

As a model problem, the linear elliptic SPDE presented in Section 2.1 was solved (this is a groundwater flow model on an L-shaped domain; see Eq. (2.2)). Three different spatial discretisations with 118, with 539, and with 2088 degrees of freedom were used. The stochastic ansatz was chosen as the polynomial chaos of second degree in 64 Gaussian random variables (2145 stochastic degrees of freedom). Hence, the discretisations had 254 110, 1 156 155, and 4 478 760 degrees of freedom.

The operator was expanded in 63 KL-terms and together with the mean matrix $\boldsymbol{K}_0$, 64 matrices $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_{63}$ were used in the sum in Eq. (4.24). The set $\mathcal{J}(\mathcal{I})$ for

the polynomial chaos expansion of the truncated KL-expansion was chosen as the polynomial chaos of second degree in the 64 basic Gaussian random variables.

The measurements in this section show the times and the parallel efficiencies for solving this problem on varying numbers of processors. First, a fixed problem size is used, but later experiments are discussed that scale the problem size with the number of processors.

As the focus is here on the parallelisation of the stochastic part of the discretisation, the parallel efficiency on N processors is defined in the following with respect to the execution time on one processor group. That is, it is defined as

$$\text{parallel efficiency} = \frac{T(1)}{T(N)N},$$

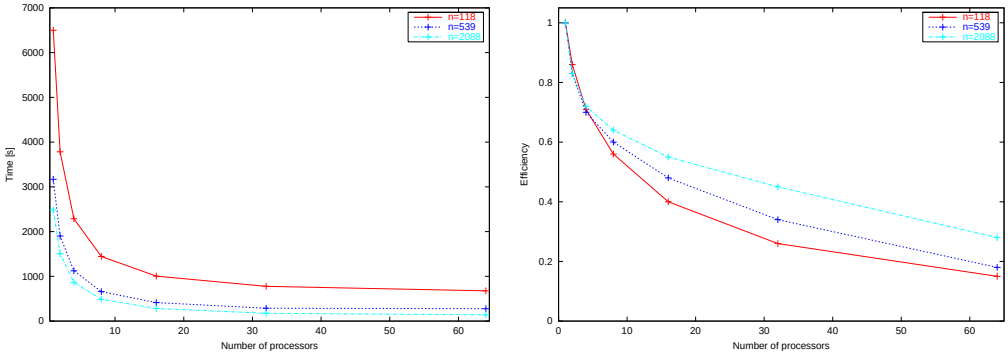
where $T(1)$ is the time required for the solution on one processor group and where $T(N)$ is the time required for solving the problem on N processor groups.

Solving the problem with the finest spatial discretisation to working precision required 13 iterations of the preconditioned block conjugate gradients solver. However, the following measurements show only the time required for one iteration of the parallel solver.

No Redundancy: First, the redundancy-free case discussed in Section 4.3.2.1 was investigated. The matrices $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_{63}$ and the block vectors were distributed over the processors as evenly as possible. The time without the communication overhead for the execution of the block matrix-vector product was seen in Eq. (4.25) to be independent of the number of processors, while the communication overhead was seen to increase quadratically with the number of processors. The time required for the preconditioning is inversely proportional to the number of processors. Hence, the parallel efficiency decreases for a growing number of processors.

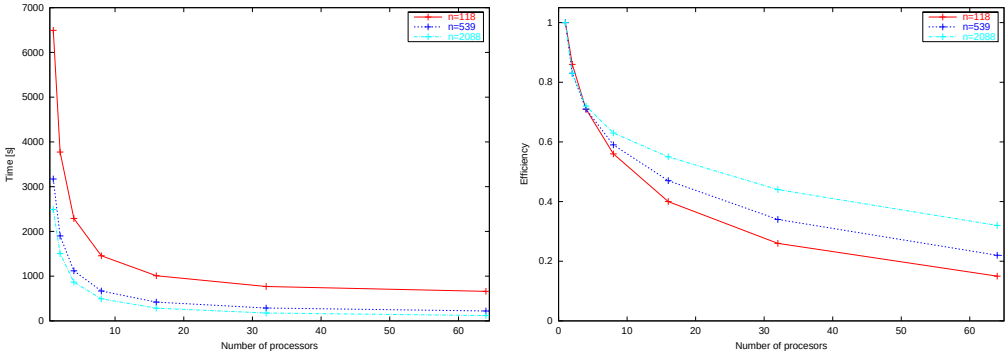
This is reflected in the timings. Fig. 4.6(a) shows the time (in seconds) that is required for one iteration of the parallel solver (one conjugate gradients iteration, including preconditioning). The parallel efficiency is shown in Fig. 4.6(b) and, as expected, the parallel efficiency decreases with an increasing number of processors for all problem sizes. The smaller spatial discretisations show a better efficiency than the finer spatial discretisations. A reason may be that for a given number of processors the communication overhead grows stronger with the size of the spatial discretisation (and hence with the amount of data being exchanged) than the savings in the preconditioner.

Distributed Block Vectors: Next, the parallelisation of Section 4.3.2.2 was applied. The block vectors were distributed redundancy-free over the processors, while all matrices $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_{63}$ were copied to each processor. According



(a) Execution time for one iteration (b) Effi ciency, no redundancy

Figure 4.6: Execution time and parallel effi ciency, No Redundancy

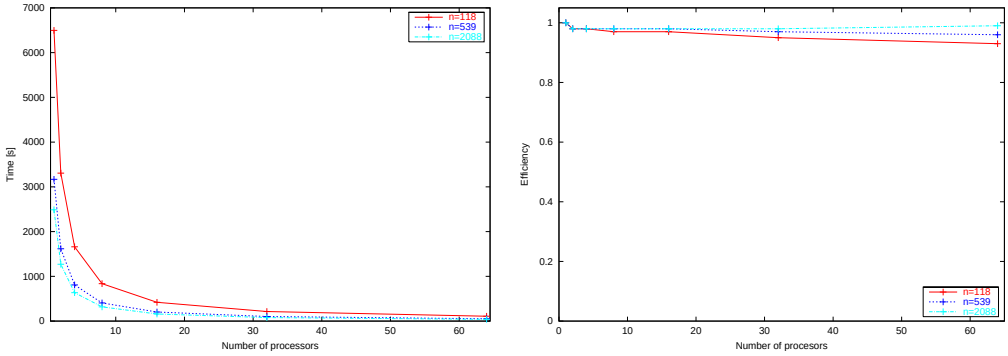


(a) Execution time for one iteration (b) Effi ciency

Figure 4.7: Execution time and parallel effi ciency, distributed block-vectors, replicated operator

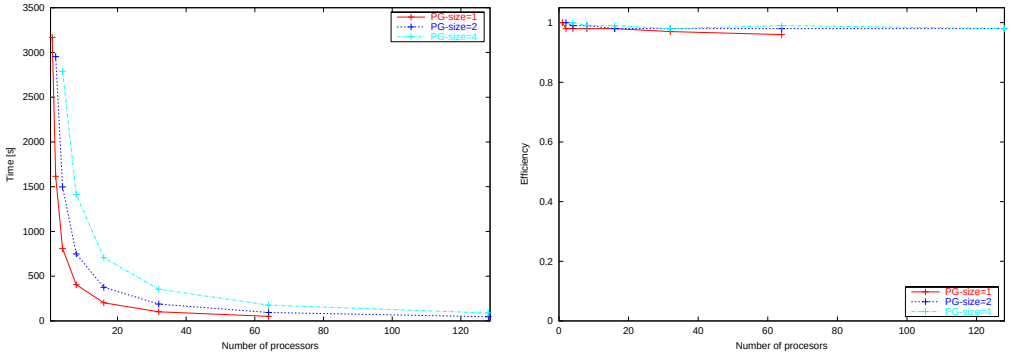
to Eq. (4.26), the time for the execution of the block matrix-vector product (ignoring the communication overhead) is again independent of the number of processors, while the communication overhead increases linearly with the number of processors. The time required for the preconditioning is inversely proportional to the number of processors. Again, the parallel efficiency decreases for a growing number of processors.

The time (in seconds) required for one iteration of the parallel solver for this parallelisation is shown in Fig. 4.7(a). As Fig. 4.7(b) shows, the parallel efficiency decreases for a growing number of processors. As expected from Eq. (4.26), the graphs are almost identical to the graphs in Fig. 4.6(a) and in Fig. 4.6(b).



(a) Execution time for one iteration (b) Effi ciency

Figure 4.8: Execution time and parallel effi ciency, distributed operator, replicated block-vectors



(a) Execution time for one iteration (b) Effi ciency with respect to the execution on one processor group

Figure 4.9: Execution time and parallel effi ciency, distributed operator, replicated block-vectors, influence of parallel deterministic code

Distributed Operator: Finally, only the operator was distributed redundancy-free over the processors, while all block-vectors were replicated on each processor.

This case was discussed in Section 4.3.2.3. According to Eq. (4.27), the total execution time without the communication overhead of the matrix vector product is inversely proportional to the number of processors, while the communication overhead increases quadratically with the number of processors.

The time required for one iteration of the parallel solver is shown in Fig. 4.8(a). As expected, an almost perfect efficiency is obtained for all spatial discretisations; see Fig. 4.8(b).

To demonstrate the influence of the size of the processor groups and of running a parallel deterministic solver, the distribution of the operator was combined with running the deterministic solver in parallel. Every instance of the deterministic solver was run on 1, 2, or 4 processor groups, and upon this fine level of parallelisation a coarser level of parallelism was added. As before, the operator was distributed and the block vectors were replicated. The resulting execution times are shown in Fig. 4.9(a) for running the problem on 1 to 128 processors.

It may be concluded from Fig. 4.9(a) that the parallel efficiency of the deterministic solver is poor. However, the development of parallel deterministic solvers is not the intent of this thesis. Hence, a simple code based on a conjugate gradients algorithm with parallel matrix-vector products was used here as deterministic code. If a better parallel solver and a spatial discretisation with more degrees of freedom are used, then the parallel efficiency of this code would be fully exploited.

As the goal here is not to investigate the efficiency of the deterministic solver and as the deterministic solver is seen as the smallest building block for the coarser levels of parallelism, the parallel efficiencies in Fig. 4.9(b) are given with respect to the solution of the SPDE when only one processor group is (consisting of 1, 2, or 4 processors) present. It is demonstrated in Fig. 4.9(b) that the coarser level of parallelism introduced by the parallel block matrix vector product and by the parallel block preconditioner have an almost optimal efficiency.

Scaled Problem Size: The measurements shown above were obtained for a fixed problem size. Timings where the problem size is scaled with the number of processors are discussed now.

The following experiments were performed: The same spatial discretisation in 2088 degrees of freedom as above was used. For an experiment on N processors the stochastic ansatz was chosen as a $N \cdot 200$-dimensional subspace of the polynomial chaos in 32 Gaussian random variables of degree four. The problem was solved on 1, 2, 4, 8, 16, 32, 64, and on 128 processors. Hence, the size of the linear problem being solved was increased from 417 600 linear equations on one processor to 53 452 800 linear equations on 128 processors.

The parallelisation was performed by replicating the operator on each processor group and by distributing the block vectors. 200 sub-vectors of each block vector were stored on each processor group. The total execution time for this problem is shown in Fig. 4.10.

Apparently, the execution time grows almost linearly with the number of processors. But to obtain the total computing time, this time must be multiplied by the number of processors. Hence, the amount of computation increases quadratically with the number of processors and with the problem size.

No statement about the efficiency of this parallelisation is made here. The reason is that statements about the parallel efficiency require to know the effort

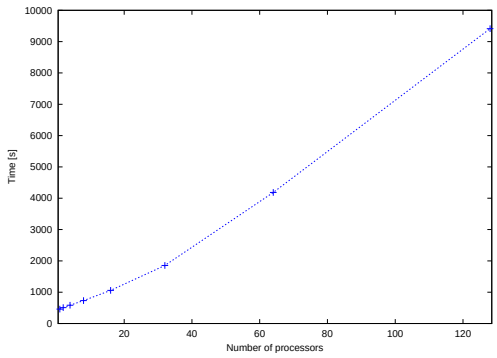


Figure 4.10: Execution time, problem size scaled with number of processors

for performing the block matrix-vector product. However, it is not clear how this effort increases with the number of unknowns. From Eq. (4.24) one might deduce that the effort for computing the block matrix-vector product grows quadratically with the number of unknowns and then this parallelisation would have a good parallel efficiency. But the block sparsity structure of the block matrix is exploited by the parallel solver and hence the effort grows slower than quadratically with the number of unknowns. As Fig. 4.1 demonstrates, the block matrix is not sparse. Hence, the effort grows faster than linearly with the number of unknowns, but it is not known how.

While no statements about the efficiency can be made for this problem, this experiment demonstrates that the parallel solver may be applied to solve problems with many millions of degrees of freedom.

4.3.7 Conclusions

Various parallelisation techniques were presented and tested. They allow to configure the parallelisation such that it is appropriate for the problem at hand: To work with problems that have a high number of spatial degrees of freedom the deterministic solvers may be a parallel program of which different instances are run in parallel on each processor group. The coarser level of parallelism use these processor groups as smallest building blocks and thus allow to use large stochastic ansatz spaces and to speed up the solution.

Because of the representation of the block matrix as a sum of tensor products, the parallel efficiency decreases if the block vectors are distributed over the processor groups. This was discussed in Section 4.3.2 and is demonstrated by the efficiency measurements in Fig. 4.6(b) and in Fig. 4.7(b). To obtain a better parallel efficiency, execution time may be traded for memory demands and the block vectors may be replicated. It was shown that almost perfect speedup is obtained if

the maximum redundancy is used for the block vectors.

It is also possible to distribute the matrices $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_l$ over the processor groups and to replicate the block vectors only a small number of times. This results in a hierarchical parallel execution: At the finest level runs a parallel deterministic code. On a coarser level each block vector group stores a copy of the block vectors and a subset of the KL-matrices $\boldsymbol{K}_0, \ldots, \boldsymbol{K}_l$. Each block vector group computes all contributions of this subset of the KL-matrices. On an even coarser level, these block vector groups run in parallel.

The efficiency measurements shown here were computed for a fixed problem size. For a problem size scaled with the number of processors only the timings were shown as it is not clear how the effort of the block matrix-vector product increases with the number of unknowns.

It was shown that the parallel solver allows to tackle very large problems. Depending on the type of parallelisation, very good parallel efficiencies were obtained and the hierarchical parallelism permits to adapt the solver to the properties of the discretisation at hand.

4.4 Solution of Nonlinear Stochastic Partial Differential Equations

This section presents solvers for nonlinear SPDEs that stem from minimisations problems (see Section 2.3.3). The discretisation Eq. (3.45) results in $n \cdot |\mathcal{I}|$ coupled equations

$$\boldsymbol{r}^{(\beta)}(\mathbf{u}) := \mathbf{E}\Big(\boldsymbol{r}\Big(\sum_{\alpha\in\mathcal{I}} \boldsymbol{u}^{(\alpha)} H_\alpha(\theta), \theta\Big) H_\beta(\theta)\Big) = 0, \qquad \forall \beta \in \mathcal{I}, \tag{4.28}$$

where $\boldsymbol{r}(\boldsymbol{u}(\theta),\theta)$ denotes the spatial semi-discretisation Eq. (3.25). Each subvector $\boldsymbol{r}^{(\alpha)}(\mathbf{u})$ has the size of the spatially discretised problem, and with the residual block vector $\mathbf{r}(\mathbf{u}) = (\ldots, \boldsymbol{r}^{(\alpha)}(\mathbf{u})^T, \ldots)^T$ this is written as the system of equations

$$\mathbf{r}(\mathbf{u}) = \mathbf{0}. \tag{4.29}$$

The evaluation of the residual and of the solution are considerable more expensive than for linear SPDEs. The block system of nonlinear equations will be solved by approximate Newton and quasi Newton methods in Section 4.4.2.

4.4.1 Evaluating the Residual

Solving Eq. (4.29) requires to evaluate the residual. The residual may be evaluated for linear SPDEs by the Kronecker product representation Eq. (4.22), but this is not directly applicable to general nonlinear SPDEs.

In [81] the residual was computed by evaluating $\mathbf{E}\left(\boldsymbol{r}(\boldsymbol{u}(\omega),\omega) H_\alpha(\omega)\right)$ for differentiable $\boldsymbol{r}$ by Eq. (A.16) as

$$\mathbf{E}\big(\boldsymbol{r}(\boldsymbol{u}(\omega),\omega) H_\alpha(\omega)\big) = \frac{1}{\sqrt{\alpha!}} \mathbf{E}\big(D_\alpha \boldsymbol{r}(\boldsymbol{u}(\omega),\omega)\big). \tag{4.30}$$

This was used to compute the expansion $\boldsymbol{r}\big(\sum_\beta \boldsymbol{u}^{(\beta)} H_\beta(\theta),\theta\big) = \sum_\alpha \boldsymbol{r}^{(\alpha)}(\mathbf{u}) H_\alpha(\theta)$. The residual was then evaluated just as shown for the linear case in Section 4.1. However, this technique cannot be used for general nonlinear SPDEs.

In principle, any high-dimensional integration technique from Section 3.3 may be used to evaluate the expectations in Eq. (4.28). The first technique that comes to mind is Monte Carlo integration. However, the following example shows that a naive Monte Carlo integration is usually not a good choice for this:

Example 4.6: As a representative example, the residual Eq. (4.28) is evaluated for a simple case with only one spatial degree of freedom. It models a spring

where the reaction force depends nonlinearly on the displacement $u \in \mathbb{R}$ and is given by the equation

$$r(u) = 0, \qquad r(u) := \kappa(u)u - 1, \quad u \in \mathbb{R}, \tag{4.31}$$

where $\kappa(u) = a + bu^2$ with random variables a and b. These are specified as nonlinear transformations $a = a(\theta_1)$, $b = b(\theta_2)$ of independent standard Gaussian random variables $\boldsymbol{\theta} = (\theta_1, \theta_2)$, such that both a and b are $\beta(1/2, 1/2)$-distributed. The displacement $u = u(\boldsymbol{\theta})$ is now a random variable satisfying

$$0 = r(u, \boldsymbol{\theta}) = \kappa(u(\boldsymbol{\theta}), \boldsymbol{\theta})u(\boldsymbol{\theta}) - 1. \tag{4.32}$$

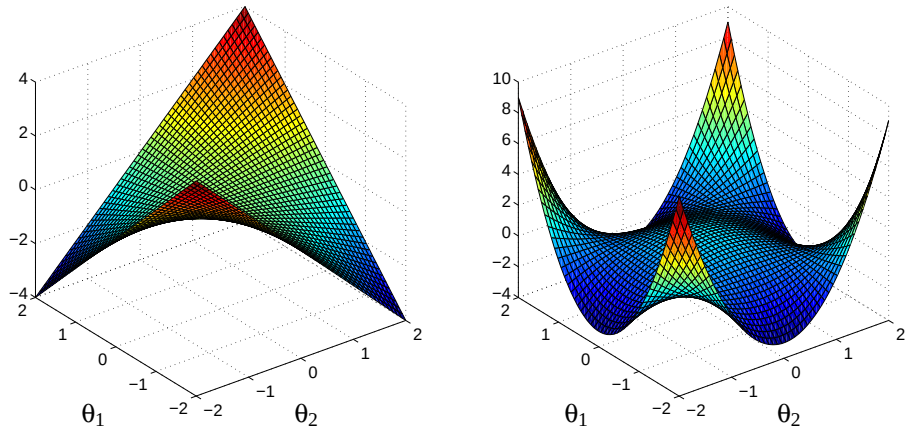
The stochastic discretisation is performed by a polynomial chaos expansion of total degree 4 in θ_1 and θ_2. Galerkin conditions are applied and the residual Eq. (3.45) is computed for a typical coefficient vector $\mathbf{u}$ by Monte Carlo integration and by quadrature. Table 4.6 shows the accuracy obtained depending

Total polynomial degree $\lvert\alpha\rvert$ of H_α	Standard-deviation of $r_\alpha(\omega)$	Monte Carlo $Z = 10^6$ Error $\cdot 10^3$	Quadrature $Z = 36$ Error $\cdot 10^3$
0	0.26	0.5	≈ 0
1	0.27	0.2	0.008
2	0.61	1.2	≈ 0
3	0.77	1.5	0.07
4	2.29	4.5	≈ 0

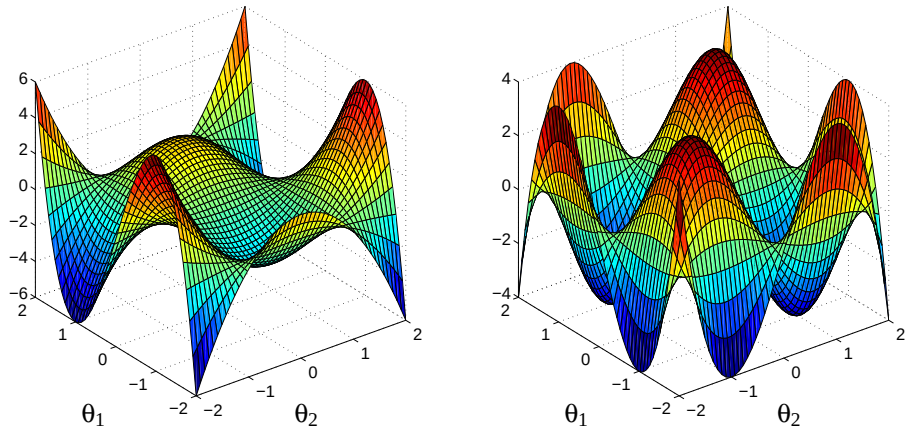
Table 4.6: Evaluation of the residual for the nonlinear stochastic spring example.

on the integration method by displaying the standard deviation of the residuum $r_\alpha(\omega) = r(\sum_\alpha u^{(\alpha)} H_\beta(\omega), \omega) H_\alpha(\omega)$ for selected α. It also shows the absolute errors for the residual components $\mathbf{E}(r_\alpha)$ for Monte Carlo integration and for Gauss-Hermite quadrature with partial polynomial exactness of order 11 (see Appendix B.3). Z is the number of integration points.

According to Table 4.6, the standard deviation of the residual components $r_\alpha(\omega) := r(\sum_\alpha u^{(\beta)} H_\beta(\omega), \omega) H_\alpha(\omega)$ grows with increasing $|\alpha|$. The reason may be that the orthogonal polynomials H_α oscillate for large $|\alpha|$, as Fig. 4.11 demonstrates. The nonlinear transformation of the oscillating ansatz $r(\sum_\beta u^{(\beta)} H_\beta(\omega), \omega)$ combined with the multiplication by an oscillating function H_α results in an integrand with high variance. As a consequence, a significant error for $\mathbf{E}(r_\alpha(\omega))$ remains in this example even for a million Monte Carlo simulations. In contrast, the quadrature errors are negligible for only few evaluations as the residual is smooth in $\boldsymbol{\theta}$.



(a) $H_{(1,1)}(\theta_1,\theta_2)$ (b) $H_{(2,2)}(\theta_1,\theta_2)$



(c) $H_{(2,3)}(\theta_1,\theta_2)$ (d) $H_{(3,3)}(\theta_1,\theta_2)$

Figure 4.11: Some Hermite Polynomials in two random variables

The same qualitative behaviour as in the previous example is observed in the numerical solution of SPDEs. Of course, the "naive" Monte Carlo method may be enhanced, for example by variance reduction techniques, but the principal problem remains.

When computing the residual for an SPDE, the numerical evaluation of the expectation in Eq. (4.28) in Z integration points requires Z evaluations of $\boldsymbol{r}(\boldsymbol{\omega}) =$

$\boldsymbol{r}\big(\sum_{\beta\in\mathcal{I}} \boldsymbol{u}^{(\beta)} H_\beta(\boldsymbol{\omega}), \boldsymbol{\omega}\big)$, where $\boldsymbol{\omega} \in \Omega^{(m)}$ denotes an integration point. This requires to evaluate many realisations $\boldsymbol{r}(\boldsymbol{\omega})$ of the residual of the spatial semi-discretisation. The deterministic code may be utilised for this in a black-box fashion. Usually, the evaluation of $\boldsymbol{r}(\boldsymbol{\omega})$ for a given $\boldsymbol{\omega} \in \Omega^{(m)}$ involves a numerical integration over the spatial domain $R \subset \mathbb{R}^d$ and the effort for this is not negligible. It is hence important to use an efficient integration rule with few integrations points.

High-dimensional quadrature techniques are used here to evaluate the residual for nonlinear SPDEs. In small stochastic dimensions, full tensor product quadrature is used, and in higher stochastic dimensions the residuals are evaluated by Smolyak quadrature; see the Appendix B.4 and the discussion in Section 3.3.3.

4.4.2 Iterative Solvers for the Nonlinear System

To solve the nonlinear system Eq. (4.29), approximate Newton or quasi-Newton methods are used [e.g. 35, 130].

Let $\mathbf{u}_j, j \in \mathbb{N}_0$ be the current guess for the solution and $\mathbf{u}_0$ the initial guess. The next approximation is obtained as $\mathbf{u}_{j+1} := \mathbf{u}_j + \Delta\mathbf{u}_j$ with a correction $\Delta\mathbf{u}_j$. Newton-like methods compute in every iteration a matrix $\mathbf{J}_j$. They differ in how they compute the matrix $\mathbf{J}_j$ used to obtain the correction as

$$\mathbf{J}_j \, \Delta\mathbf{u}_j = -\mathbf{r}(\mathbf{u}_j). \tag{4.33}$$

Newton method: For the Newton method, $\mathbf{J}_j$ is the Jacobian of the residual, $\mathbf{J}_j := D_{\mathbf{u}}\mathbf{r}(\mathbf{u}_j)$. This is a block matrix $(D_{\mathbf{u}}\mathbf{r}(\mathbf{u}_n))_{\alpha,\beta\in\mathcal{I}}$, where every block has the size of the spatial discretisation.

Example 4.7: The Jacobian of our nonlinear SPDE Eq. (2.20) is

$$\begin{aligned}
(D_{\mathbf{u}}\mathbf{r}(\mathbf{u}))_{\alpha,\beta} &= D_{\boldsymbol{u}_\beta}\Big(\mathbf{r}_\alpha(\mathbf{u})\Big) && (4.34)\\
&= D_{\boldsymbol{u}_\beta} \sum_{\beta'} \int_R \nabla \boldsymbol{N}(x) \kappa_{\alpha,\beta',\mathbf{u}}(x) \nabla(\boldsymbol{N}(x))^T \, dx \, \boldsymbol{u}^{(\beta')} && (4.35)\\
&= \int_R \nabla \boldsymbol{N}(x) \kappa_{\alpha,\beta,\mathbf{u}}(x) \nabla(\boldsymbol{N}(x))^T \, dx && (4.36)\\
&\quad + \sum_{\beta'} \int_R \nabla \boldsymbol{N}(x) \big[D_{\boldsymbol{u}_\beta} \kappa_{\alpha,\beta',\mathbf{u}}(x)\big] \nabla(\boldsymbol{N}(x))^T \, dx \, \boldsymbol{u}_{\beta'}, && (4.37)
\end{aligned}$$

where

$$\kappa_{\alpha,\beta',\mathbf{u}}(x) := \mathbf{E}\Big(H_\alpha(\theta) H'_\beta(\theta) \kappa\Big(\sum_{\beta''} \boldsymbol{N}(x)^T H_{\beta''}(\theta) \boldsymbol{u}^{(\beta'')}, \theta\Big)\Big).$$

Each term in the Jacobian looks exactly like a normal finite element stiffness matrix with material properties given by expectations. The Jacobian may hence be computed by the deterministic code.

Every iteration of the Newton method involves the large system of block equations Eq. (4.33). The solvers for linear block systems that were developed in Section 4.2 expect the block matrix to be represented as in Eq. (4.13) as a sum of Kronecker products $\sum_\gamma \Delta^{(\gamma)} \otimes \boldsymbol{K}^{(\gamma)}$ with appropriate matrices $\boldsymbol{K}^{(\gamma)}$, where $\Delta^{(\gamma)}$ is defined in Eq. (4.12). By expanding $\kappa_{\alpha,\beta',\mathbf{u}}(x)$ in polynomial chaos, the first part of the Jacobian, Eq. (4.36), may be written in this form, but not the second part, Eq. (4.37).

The second part is therefore neglected, resulting in an approximate Newton method with inexact Jacobian, where any of the linear solvers of Section 4.2 may be used to compute the correction.

Quasi-Newton Method: This nested iteration (a Newton method with an iterative linear solver) may be avoided by employing a quasi-Newton method with line-searches [114, 166].

The nonlinear elliptic SPDE of Section 2.3.3 may be seen as minimisation problem. Hence, the BFGS method (named after Broyden, Fletcher, Goldfarb and Shannon) may be a good choice [see e.g. 35, 114, 130, 166].

The BFGS method computes the inverse iteration matrix $\mathbf{J}_j^{-1}$ as a rank-two update of the previous inverse iteration matrix $\mathbf{J}_{j-1}^{-1}$. This allows to compute the correction of the current block-vector iterate $\mathbf{u}_j$ as

$$\mathbf{u}_{j+1} - \mathbf{u}_j = -\mathbf{J}_j^{-1}\mathbf{r}(\mathbf{u}_j), \tag{4.38}$$

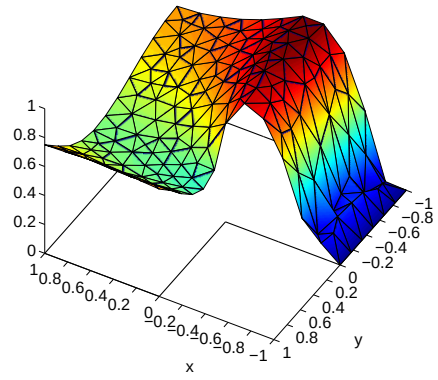
where

$$\begin{aligned}\mathbf{J}_j^{-1} &= \mathbf{J}_{j-1}^{-1} + \mathbf{v}_j\mathbf{v}_j^T + \mathbf{w}_j\mathbf{w}_j^T \\ &= \mathbf{J}_0^{-1} + \sum_{k=1}^{j} \left(\mathbf{v}_k\mathbf{v}_k^T + \mathbf{w}_k\mathbf{w}_k^T\right).\end{aligned}$$

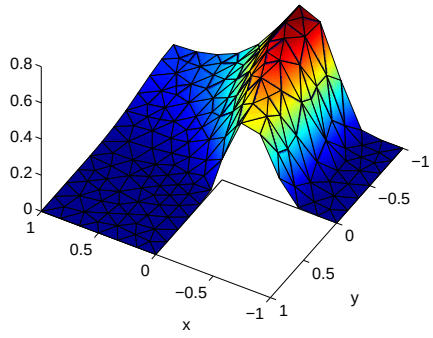
The vectors $\mathbf{v}_1, \ldots, \mathbf{v}_j$ and $\mathbf{w}_1, \ldots, \mathbf{w}_j$ are obtained as by-products of the previous iterations [35, 114, 130, 166] and the updates in Eq. (4.38) are stored as single vectors without actually computing the tensor products.

The inverse of an initial matrix $\mathbf{J}_0$ is required, which may be seen as a preconditioner [35, 130]. It should be chosen such that $\mathbf{J}_0 \approx D_\mathbf{u}\mathbf{r}(\mathbf{u}_0)$ and such that linear equations involving $\mathbf{J}_0$ may easily be solved. Both goals are achieved here by selecting $\mathbf{J}_0 = \mathbf{P}$, where $\mathbf{P}$ is defined in Eq. (4.23). It is the same preconditioner that was used for the linear solvers. The preconditioner is block-diagonal and each block in the diagonal is the Jacobian of the deterministic system obtained by replacing the stochastic fields by their mean values.

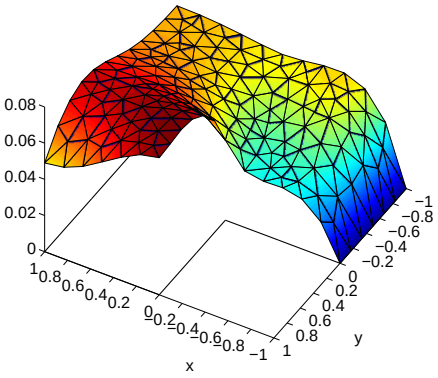
4.4.3 Numerical Experiments



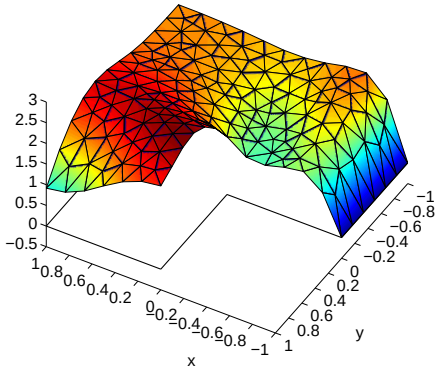
(a) Error $\cdot 10^4$ of $\mathbf{E}\left(u_{galerk}\right)$



(b) Prob$\{u_{galerk}(x) > 3.25\}$



(c) $u^{(\alpha)}_{direct} = \mathbf{E}\left(u(\omega)H_\alpha\right)$



(d) Error $\cdot 10^4$ in $u^{(\alpha)}_{galerk}$

Figure 4.12: Solutions and errors, direct projection method and Galerkin method

Numerical experiments were performed to test the nonlinear solver. These results were published in [81, 84, 87].

For the same groundwater-flow problem that was solved in Section 3.3.4 by direct numerical integration, the solution was also obtained by the stochastic Galerkin method of Section 3.4.2 and by the orthogonal projection method of Sec-

tion 3.4.3. See Section 3.3.4 for a description of the parameters and of the boundary conditions that were used.

The stochastic ansatz was chosen as the 6-dimensional polynomial chaos of degree 2 (28 stochastic functions). A spatial discretisation in 170 degrees of freedom was performed, totalling $4,760$ nonlinear equations. The BFGS solver required 19 iterations, and as the first iterations required line-searches, the residual had to be evaluated 24 times. The residual was integrated by the 5-stage Smolyak quadrature S_5^6 in $Z = 1,820$ integration points. As the evaluation in each integration point required one integration in the spatial dimension, $43,680$ integrations over the spatial domain were performed.

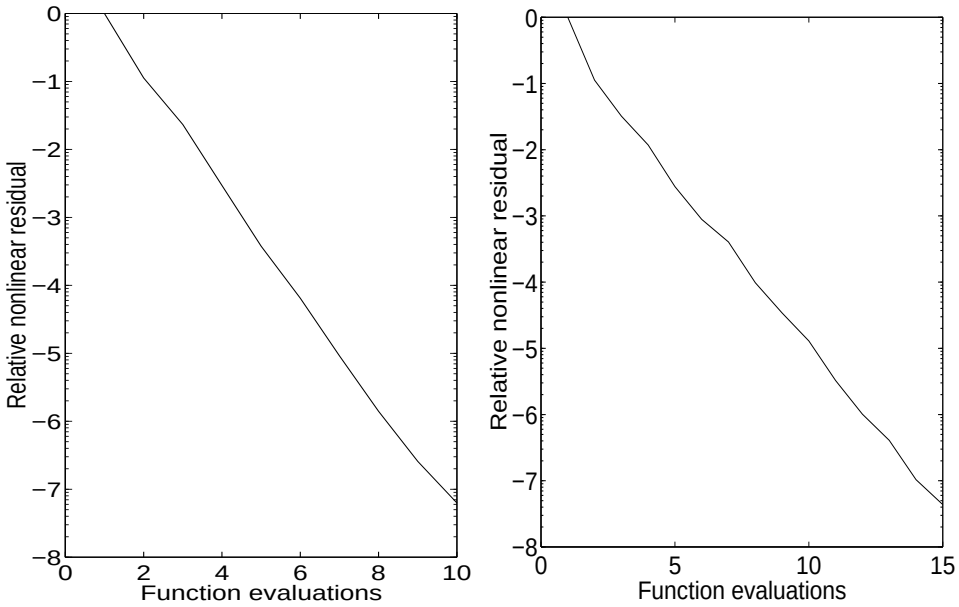
For the BFGS-solver, it was important to choose a good preconditioner $\mathbf{H}_0$ for the nonlinear system. Here the Block-Jacobi solver for the linear system $D_{\mathbf{u}}\mathbf{r}(0)$ was used. As the BFGS-solver required 19 iterations and as 28 stochastic degrees of freedom were used, 532 linear systems with the size of the spatial discretisation were solved.

To assess the validity of the results, the mean of the solution obtained by the Galerkin scheme was compared to the results computed in Section 3.3.4 by direct Monte Carlo and by direct Smolyak integration. The error is shown in Fig. 4.12(b) and it is small in the "eye-ball" norm.

To compare the orthogonal projection method of Section 3.4.2 to the Galerkin method, a reference solution for the block-vector $\mathbf{u}$ was computed by orthogonal projection. The orthogonal projections were evaluated by the 6-stage Smolyak quadrature formula S_6^6 requiring $6,188$ integration points. In each integration point the deterministic problem was solved for the material parameters associated with that point. Fig. 4.12(c) shows the component for $\alpha = (0,0,0,1,0,0)$ of the reference solution and Fig. 4.12(d) shows the error of the Galerkin approximation with respect to this component. Again, the error is small and the error of the other components that are not shown here is of the same order of magnitude.

Note that the orthogonal projection method required significantly more effort for the solution than the BFGS-solver: While the orthogonal projection method required to solve 6,188 nonlinear systems, each with the size of the spatial discretisation, the BFGS-solver required to solve only 532 linear systems, each with the size of the spatial discretisation. The most significant part of work for the BFGS-solver were the $43,680$ integrations over the spatial domain required in evaluating the residual.

To demonstrate the advantages of the resulting response surface representation, the probability $p_{u_0}(x) = \text{Prob}\{u^{h,\mathcal{I}}(x) > 3.25\}$ was computed. For this, a naive Monte Carlo simulation with $100,000$ samples was used. As realisations of the solution can be obtained at negligible costs from the response surface $u_{galerk}(x,\omega)$, this computation is much cheaper than a direct application of Monte Carlo techniques. The result is shown in Fig. 4.12(b).



(a) Newton with inexact Jacobian (b) BFGS-method with line-searches

Figure 4.13: Nonlinear SPDE, Decrease of residual, semi-logarithmic plots.

A comparison of the approximate Newton solver and of the BFGS solver was also performed [81]. The following results were obtained for the same SPDE but with slightly different parameters: approximately twice the number of equations were solved and the hydraulic conductivity was lognormally distributed.

The behaviour of both nonlinear solvers is shown in Fig. 4.13, where the decrease of $\|\mathbf{r}(\mathbf{u}_j)\|$ is plotted over the number of iterations j. The approximate Newton method shows the expected linear convergence behaviour. Quasi-Newton-methods are super-linearly convergent close to the solution, but the convergence of the BFGS-method that was observed here seems linear, so the region of super-linear convergence was not reached. The approximate Newton method used less iterations than the BFGS solver, but in toto, the latter was faster as it did not require to run the linear solver in each iteration.

Both nonlinear iterative solvers were applied with line searches with a loose tolerance [114, 166].

In the approximate Newton method with the Jacobian replaced by Eq. (4.36), the linear system arising in each step was solved by a conjugate gradient method preconditioned by a block JOR solver. The solution of the linear system $D_{\mathbf{u}}\mathbf{r}(0)$ was used as initial guess. This choice was in many experiments found to be a good initial guess.

4.4.4 Conclusions

As the numerical experiments show, the solution computed by the orthogonal (non-intrusive) projection method of Section 3.4.2 and the solution obtained by the stochastic Galerkin-method of Section 3.4.3 agree well. Further, these results match with the results obtained by the direct numerical integration methods of Section 3.3 where Monte Carlo and Smolyak integration methods were used.

According to this and according to experiences made with other numerical experiments, both the stochastic Galerkin method with a BFGS-solver and the orthogonal projection method are promising techniques for the solution of nonlinear elliptic SPDEs that stem from minimisation problems.

The Galerkin method obtains the solution as a large coupled system of nonlinear block-equations while the orthogonal projection obtains the solution by solving many uncoupled problems and by integrating the realisations of the solution over the stochastic space. It may hence be expected that the orthogonal projection method is more robust than the Galerkin method.

In the experiments performed here, the Galerkin method obtained the solution significantly faster than the orthogonal projection technique: the latter technique requires to solve a nonlinear PDE for every integration point, while the Galerkin method employs high-dimensional integration-methods only to integrate the residual. The stochastic Galerkin method solves a nonlinear PDE only to obtain an initial guess and employs a (constant) block-diagonal linear solver for the preconditioning.

Which of the considered techniques (direct integration method, orthogonal projection method, Galerkin method) is the most efficient depends on the stochastic dimensions, on the spatial discretisation, and on properties of the solution, like its variance and its smoothness in the stochastic space. A systematic analysis of this would be desirable, but has not been performed yet.

Solving nonlinear SPDEs requires a high effort. The solution of the resulting large system of nonlinear equations may be performed by a approximate Newton method using the linear block-solvers discussed in Section 4.1, thus exploiting the special structure of the resulting linear equations.

This nested iteration was avoided by employing quasi-Newton methods for the solution. If the problem at hand stems from a minimisation problem, then the BFGS method may be a good choice. It was shown that the BFGS-method can be applied more easily than the approximate Newton method as it does not require the Jacobian. In the numerical experiments the BFGS-technique showed good convergence and it was faster than the Newton method.

By employing the BFGS-method with line searches, general nonlinear elliptic SPDEs of gradient type may be solved. Only a routine for evaluating the residual of the SPDE and the deterministic solver are required. It was shown that

Monte Carlo methods may not be well-suited for evaluating the residual, and high-dimensional quadrature routines were proposed and implemented as an efficient alternative.

As a conclusion, in combining the BFGS-method with efficient high-dimensional integration routines for computing the residual, a general-purpose solver is obtained that may be used to solve general gradient type nonlinear elliptic SPDEs.

4.5 An Adaptive Solver

Solutions of SPDEs were obtained in the previous sections in given ansatz-spaces. This section discusses an adaptive solver.

Various techniques for the à posteriori error estimation, for the refinement of solutions, or for the computation of sensitivities of the system were proposed and implemented; see [53, 96] and in particular [136] and the references therein. These techniques usually employ heuristic arguments or compute sensitivities as derivatives of the answer with respect to the independent random variables $\theta = (\theta_1, \ldots, \theta_m)$. These sensitivities are valid in finite-dimensional stochastic spaces, but in general stochastic spaces an infinite-dimensional calculus must be used, e.g. the Malliavin-calculus [100]. It is hence not obvious whether sensitivities computed by taking derivatives with respect to the independent random variables are stable approximations of sensitivities of the original SPDE.

This section discusses a goal-oriented approach for the adaptive solution that may also by used for an à posteriori error estimation and for the computation of sensitivities. It is based on the observation that usually not the complete solution of the SPDE is of interest but properties like its mean, its variance, or its probability to satisfy some property. Thus, usually functionals of the solution are computed. This permits a goal-oriented approach: the solution of a problem dual to the original one may be interpreted as the sensitivity of the functional [89, 101] and this may be utilised to refine the ansatz space [145].

This is presented here for a linear stationary SPDE $Au = f$ on the domain $R \subset \mathbb{R}^d$. As in Section 2.3.1, A is a linear elliptic operator $A\colon V \otimes (S) \to (V \otimes (S))^*$ on a Hilbert space $V \otimes (S)$. For example, for the SPDE Eq. (2.13), one would choose $V = \mathring{H}^1(R)$ and $(S) = L^2(\Omega)$.

As in Section 3.3, the functional of interest is called $s\colon V \otimes (S) \to \mathbb{R}$. Only continuous linear functionals are considered. For example, to compute the mean of the solution integrated over some region $M \subset R$, the functional is chosen as

$$s(u) := \int_M \mathbf{E}\,(u(x,\omega))\,dx. \tag{4.39}$$

The technique is applied here successfully to a linear SPDE, but the solution of the dual problem introduces a considerable overhead. Hence, this approach

would make more sense for an instationary problem or for a nonlinear problem. In principle, this technique may also be used for such problems and for nonlinear functionals [101], but this was not tested here.

4.5.1 Linear Adaptivity

It is assumed here that the SPDE of interest is solved as in Section 3.4.3 by a Galerkin approximation in a space $V^h \otimes (S)^{\mathcal{I}} \subset V \otimes (S)$. Recall that V^h denotes a space of finite element shape functions with mesh-size $h > 0$ and that $(S)^{\mathcal{I}}$ denotes a set of stochastic ansatz functions identified by the finite set of multi-indices $\mathcal{I}$. The discrete solution $u^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}}$ is thus defined by

$$\left\langle\left\langle Au^{h,\mathcal{I}}, v^{h,\mathcal{I}}\right\rangle\right\rangle = \left\langle\left\langle f, v^{h,\mathcal{I}}\right\rangle\right\rangle, \qquad \text{for all } v^{h,\mathcal{I}} \in V^h \otimes (S)^{\mathcal{I}},$$

where $\langle\langle\cdot,\cdot\rangle\rangle$ is the duality paring between $(V \otimes (S))^* = V^* \otimes (S)^*$ and $V \otimes (S)$.

An approximate value for the functional of interest can now be computed from the approximate solution as $s(u^{h,\mathcal{I}})$. As the functional considered is bounded and linear and as $V^h \otimes (S)^{\mathcal{I}}$ is a Hilbert space, the functional may be represented by an element $\psi \in V^h \otimes (S)^{\mathcal{I}}$ as

$$s(v) = (v, \psi) \qquad \forall v \in V \otimes (S). \tag{4.40}$$

For example, the functional in Eq. (4.39) is represented by

$$\psi(x, \omega) := \chi_M(x), \tag{4.41}$$

where χ_M denotes the characteristic function of the region $M \subset R \subset \mathbb{R}^d$.

From Eq. (4.40) and from solving a problem dual to the original one, the sensitivity of the function with respect to the SPDE may be computed. For this, consider the adjoint operator A^* defined as

$$\langle\langle Au, v^*\rangle\rangle = \langle\langle u, A^*v^*\rangle\rangle, \qquad \forall u \in V \otimes (S), v^* \in (V \otimes (S))^*. \tag{4.42}$$

By solving the dual problem

$$A^*u^* = \psi, \tag{4.43}$$

a dual solution $u^* \in (V \otimes (S))^*$ is obtained that may be interpreted as the sensitivity of the functional with respect to the operator; see Marchuk's monograph [101] for details.

With this dual solution the error of the functional can be estimated: If u is the exact solution, then the error in the functional value is

$$s(u) - s(u^{h,\mathcal{I}}) = \left(u - u^{h,\mathcal{I}}, \psi\right) = \left(u - u^{h,\mathcal{I}}, A^* u^*\right) \tag{4.44}$$

$$= \left(A(u - u^{h,\mathcal{I}}), u^*\right) \tag{4.45}$$

$$= \left(f - Au^{h,\mathcal{I}}, u^*\right) \tag{4.46}$$

$$= (r, u^*), \tag{4.47}$$

where $r = f - Au^{h,\mathcal{I}}$ is the residual.

By evaluating the residual, Eq. (4.47) may be used to compute an à posteriori error estimate of the functional value, e.g. as $|s(u) - s(u^{h,\mathcal{I}})| \leq \|r\| \cdot \|u^*\|$. However, while Pierce and Giles [140] applied this approach with success to PDEs, experiments performed for SPDEs were not successful here as the error estimates that were obtained had the same order of magnitude as the value of the functional.

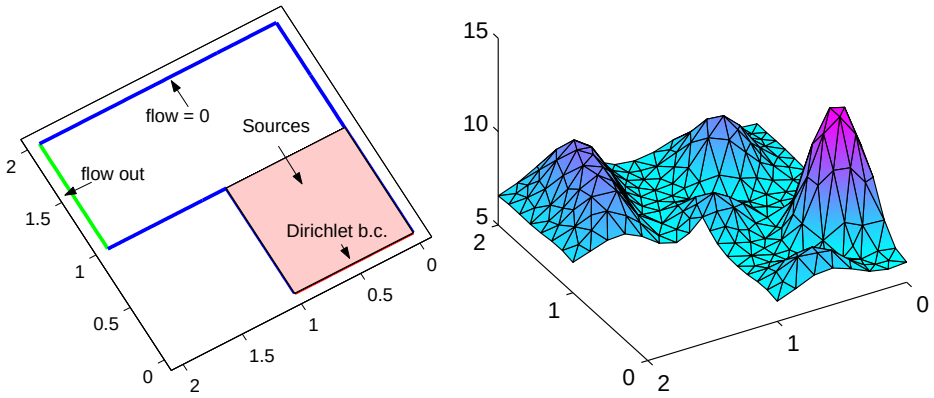
This sensitivity was successfully utilised in structural and fluid problems to estimate the error of the functional and to implement an adaptive scheme; e.g. see [89, 101, 140, 145] and the references therein. The same ideas may be applied to our SPDE. To allow the use of the deterministic code for the spatial discretisation, the refinement and coarsening is performed only in the stochastic dimensions. The adaptive scheme for refining the ansatz space is implemented here as follows:

- An approximate solution $u^{h,\mathcal{I}}$ of the SPDE and an approximate solution $u^{h',\mathcal{I}',*}$ of the dual problem are computed by the stochastic Galerkin method using the solvers discussed in the previous sections. Here, $u^{h,\mathcal{I}}$ and $u^{h',\mathcal{I}',*}$ need to be taken from different same ansatz spaces to obtain a meaningful estimate in Eq. (4.47) as otherwise $\left\langle\left\langle f - Au^{h,\mathcal{I}}, u^{h',\mathcal{I}',*}\right\rangle\right\rangle = 0$ due to the Galerkin orthogonality.

- The stochastic coarsening is performed by projecting the sensitivity $u^{h',\mathcal{I}',*}$ onto the one-dimensional subspaces spanned by each stochastic ansatz function and by then taking the scalar product with the residual. Only stochastic ansatz functions are retained for which this product is large.

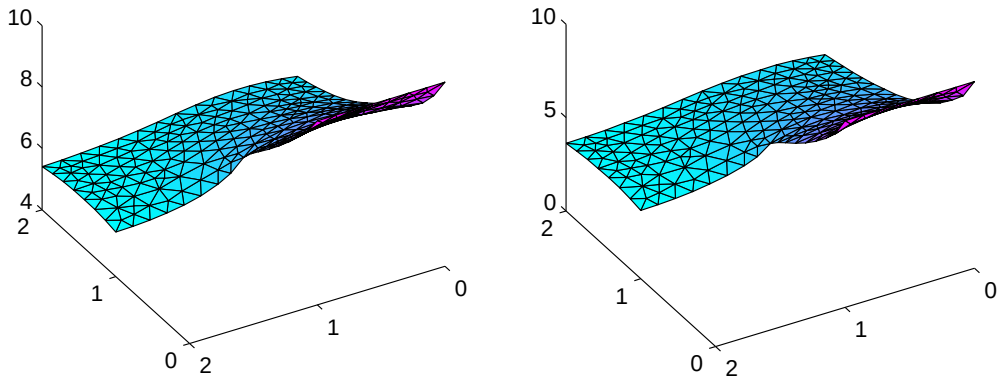
  That is, if H_α is in $(S)^{\mathcal{I}}$, then $\left\langle\left\langle f - Au^{h,\mathcal{I}}, P_\alpha u^{h',\mathcal{I}',*}\right\rangle\right\rangle$ is computed, where P_α denotes the projection onto H_α. If the result is small, then H_α is removed from the stochastic ansatz space. Experiments show that this procedure is successful at detecting unimportant ansatz functions.

- In contrast to the situation for PDEs, where adaptive refinement strategies may be based on geometrical considerations and where local error indicators inside the finite elements may be employed, it is not obvious how to



(a) Geometry (b) Realisation of κ

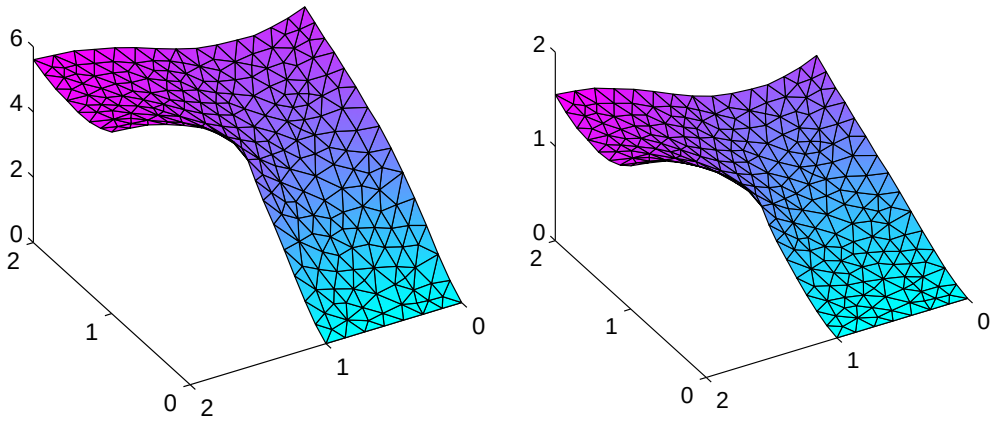
Figure 4.14: Geometry and Realisation of $\kappa(x, \omega)$.



(a) Realisation of u for κ in Fig. 4.14. (b) Mean of the reference solution.

Figure 4.15: Realisation and Mean of the Reference Solution.

refine our orthogonal stochastic ansatz. A heuristic approach is taken here: To test whether or not to include a stochastic function H_α in the ansatz, the projection of the residual onto H_α is considered. If this projection is large, then H_α is added to the stochastic ansatz.



(a) Variance of reference solution.

(b) C.O.V. of reference solution.

Figure 4.16: Variance and C.O.V. of reference solution.

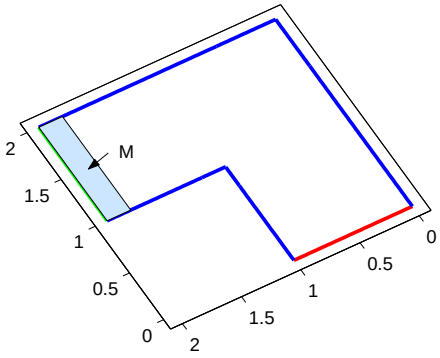
4.5.2 Experiments

To test the adaptive scheme, the linear stochastic groundwater flow problem $-\nabla \cdot (\kappa(x,\omega)\nabla u(x,\omega)) = f(x,\omega)$ was solved [82] on the spatial domain shown in Fig. 4.14. The green boundary denotes a flow-out condition, the blue boundary visualises no-flow conditions, and Dirichlet condition are applied at the red boundary. The reddish area shows the sources. The stochastic conductivity $\kappa(x,\omega)$ was modelled as a lognormal field, $\kappa(x,\omega) = \exp(\gamma(x,\omega))$, where the covariance function of the underlying Gaussian κ field was chosen as $\mathrm{cov}_\kappa(x,y) = c_1 exp(-\|x-y\|/\lambda)$, where $\lambda = 0.2$ and where c_1 was chosen such that the coefficient of variance of κ was 30%. A realisation of κ is shown in Fig. 4.14.

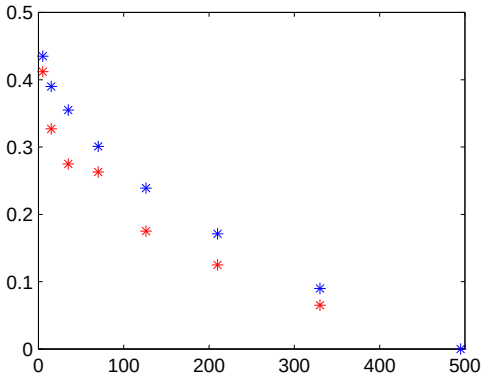
The realisation of the solution u belonging to this realisation of κ is shown in Fig. 4.15. Even though the realisation of the stochastic conductivity varies strongly, the realisation of the hydraulic head is smooth. This is because the SPDE is elliptic and models an equilibrium problem. Other realisations are also smooth, but the individual realisations differ strongly, as may be concluded from the variance of the solution shown in Fig. 4.16.

A reference solution was computed and the errors were evaluated with respect to this reference solution. For the reference solution, 254 spatial ansatz functions and 495 stochastic ansatz functions (polynomial chaos of degree four in eight mutually independent Gaussian random variables) were used, resulting in a total of 125.730 degrees of freedom. The mean, the variance, and the coefficient of variance of this reference solution are displayed in Fig. 4.15 and Fig. 4.16.

The region M over which the mean hydraulic head in the definition of the functional Eq. (4.39) is integrated is shown in blue in Fig. 4.17. The same figure



(b) Region M (in blue).



(b) Relative error plotted against number of stochastic ansatz functions.

Figure 4.17: The region M and relative errors.

shows the relative errors of the functional with respect to the reference solution. For the computation of these errors, the spatial ansatz was kept constant; only the stochastic ansatz space was modified (the number of independent random variables used in the ansatz was varied). The abscissa shows the number of stochastic ansatz functions and the ordinate shows the relative error. The blue crosses display the relative error with respect to the reference solution before the adaptation and the red crosses give the (always smaller) error obtained for the same number of stochastic ansatz functions after executing the refinement algorithm once. The relative error of the rightmost measurement is zero as it denotes the reference solution.

4.5.3 Conclusions

By solving a problem dual to the original one, a measure of sensitivity was obtained for the functional. This was used in an adaptive method to coarsen and refine the stochastic ansatz space.

The results in Fig. 4.17 show that the adaptive scheme was successful for the chosen example. For every experiment that was performed, the refinement technique obtained an ansatz space that decreased the error but had the same size as the original ansatz space. A dual problem had to be solved for the refinement and hence the method outlined here is about three times as expensive than the direct solution of the SPDE (it requires to solve the SPDE, to solve the dual problem, and then to solve the SPDE with the refined ansatz).

The experience with other experiments shows that this technique is good at coarsening the ansatz space. However, in contrast to refinement strategies for finite element spaces, there is no geometrical structure in the probability space

and there are no obvious neighbourhood relations between the polynomial chaos ansatz functions. Hence, the refinement strategy is more or less heuristic.

This approach may be better suited for nonlinear and instationary problems as the effort of solving the SPDE is much higher there. An extension of this approach to instationary and nonlinear SPDEs and to nonlinear functions may be performed similarly as for deterministic PDEs [101], but this was not implemented here.

Adaptivity was implemented here only in the stochastic dimensions, but it seems straightforward to use these ideas for space-stochastic adaptivity where both the spatial and stochastic ansatz space are refined.

Chapter 5

A General Purpose Software for Stochastic Finite Elements

The numerical techniques discussed in the previous chapters were implemented in a general-purpose software for stochastic finite elements, StoFEL ("Stochastic Finite Element Library"). This chapter discusses design aspects and the coupling with existing simulation codes.

5.1 Design and Implementation

The following exposition does not discuss all aspects of StoFEL. The focus is on the coupling with solvers for the spatial discretisation.

5.1.1 Design Goals and Code Coupling Considerations

The StoFEL library was designed to be a general-purpose framework for stochastic finite elements. A key design-goal was to allow the introduction of stochastic uncertainties into existing simulation codes without modifying these. These simulation codes will be called the *deterministic code* or the *deterministic solver* in the following.

Most simulation codes that are in use today were developed for physical models with deterministic parameters. StoFEL reuses this software for the spatial discretisation. Reusing existing software has many benefits: it allows to use state-of-the-art software for the spatial discretisation and for the solution and it reduces maintenance and development costs.

A disadvantage of interfacing to existing software is that this may result in a significant communication overhead. Similar problems occur in general purpose software for reliability computations. An overview of such approaches for

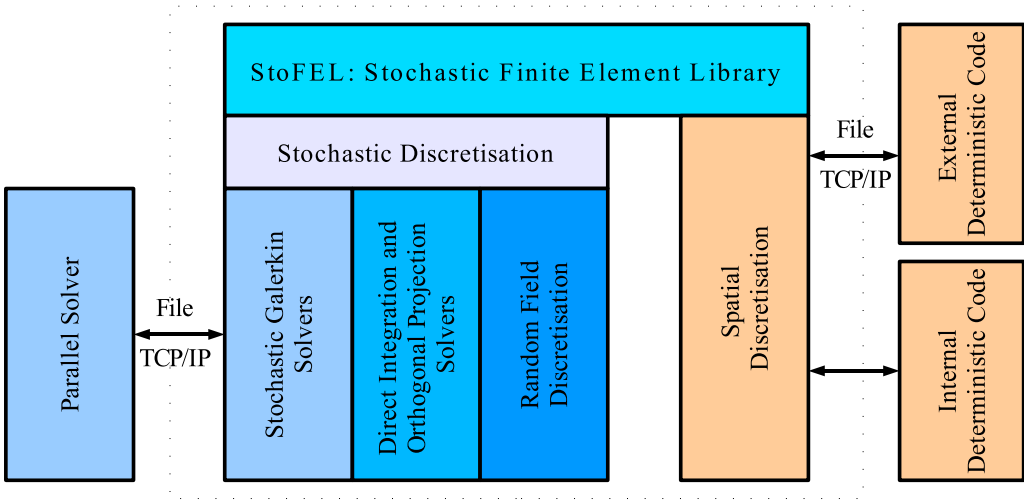


Figure 5.1: Module structure of the StoFEL-library.

stochastic structural analysis is given in [19]. There it is argued that probability integrators interfacing to external software [e.g. 17, 98] are suited only for systems with few random parameters because of the involved communication overhead. It is further argued in [19] that systems with many random parameters, e.g. systems in which random fields occur, usually need to be solved by integrated structural stochastic analysis software [e.g. 20, 21, 59] providing its own routines for the spatial discretisations to reduce the communication overhead.

The same situation is encountered here: Much communication with the deterministic code is required by most numerical techniques presented in the previous chapters. While the communication required for solving a linear SPDE may be reduced significantly by the techniques developed in Chapter 4, no such reduction seems to be possible for general nonlinear SPDEs.

Hence, coupling to an external deterministic code with non-negligible communication costs may be feasible for linear SPDEs, but it may be unfavourable for nonlinear SPDEs. For example, the coupling with ANSYS discussed in Section 5.2.2 leads to significant communication costs.

The solution techniques presented here perform better if the coupling with the deterministic code leads to negligible communication costs. This may be the case if the deterministic code resides in the same address space as StoFEL. For example, this is the case for the coupling with FEMLAB (see Section 5.2.3).

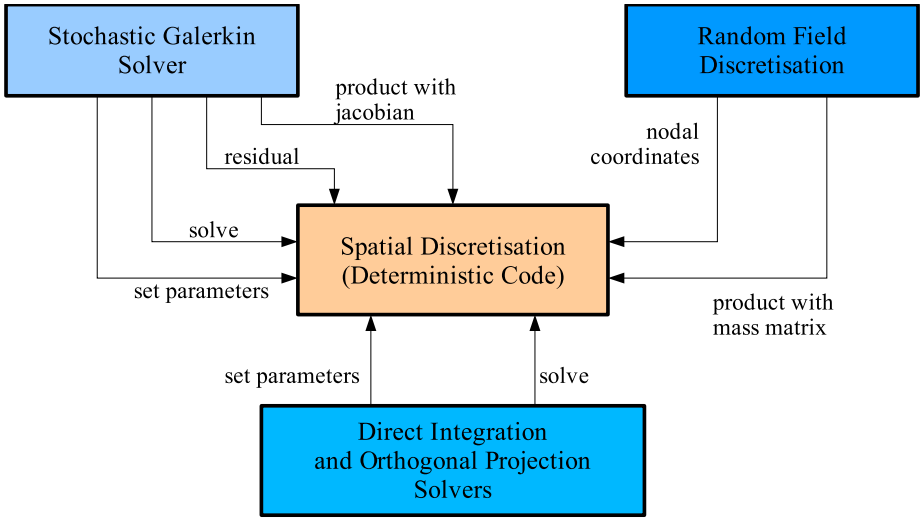


Figure 5.2: Interactions with the Deterministic Code.

5.1.2 The Overall Design

With no special application in mind, the StoFEL library had to be general, modular and extensible. Hence, an object-oriented design and implementation were chosen. Fig. 5.1 shows the overall module structure of StoFEL, which consists of three modules for the stochastic discretisation, and of one module for the spatial discretisation.

The three modules for the stochastic discretisation implement the random field discretisation, the direct integration methods, and the stochastic Galerkin methods discussed in the previous chapters. They are discussed briefly in Section 5.1.4.

The spatial discretisation module is the focus of this chapter and will be discussed in Section 5.1.3. It implements the semi-discretisation of Section 3.2 by interfacing to the deterministic code. The interface for the spatial discretisation is contained in a few abstract classes that may be sub-classed to integrate existing simulation software. Communications with the deterministic code are performed only via these classes in a way that is transparent to the other modules. Hence, any communication technique may be used for the coupling. This is indicated in Fig. 5.1 by distinguishing between internal deterministic codes (residing in the same address space as StoFEL), and external deterministic codes that may be connected, e.g. via file exchange or via network communication.

5.1.3 The Spatial Discretisation

Only few assumptions were made about the spatial discretisation to obtain a generally applicable library. The arrows in Figure 5.2 show what features of the deterministic solver are required by the numerical techniques. The arrows also visualise how the spatial discretisation module collaborates with the modules for the stochastic discretisation.

Figure 5.2 shows that the random field discretisation needs to obtain the nodal coordinates from the deterministic code and that it requires products with the mass matrix. The nodal coordinates are required for interpolating functions on the spatial domain in the FE base, for example, the mean and the covariance function of random fields. Matrix-vector products with the mass matrix are used in the discrete Karhunen–Loève expansion (see Section 3.1.3). If the deterministic code is not an FE code, then the concept of a mass matrix may not be appropriate and then the identity matrix may be used instead. In this case, the random field discretisation yields not the KL-expansion, but a principal component analysis of the random field.

The direct integration techniques (see Section 3.3) and the orthogonal projection method (see Section 3.4.2) integrate functionals of the solution over the probability space. Many different realisations of the solution must be computed for this. Solving a realisation of the SPDE is visualised in Fig. 5.2 as two steps: First, a realisation of the SPDE is obtained by assigning realisations to its parameters in the "set parameters" routine. Then the deterministic solver is called in the "solve" method.

Linear SPDEs discretised by Galerkin methods are solved by the iterative solvers of Section 4.2. These require the computation of matrix-vector products with the block system matrix, where the efficient representations for the block system matrix of Section 4.1 are exploited. Evaluating the block matrix-vector product requires to compute matrix-vector products with the system matrices for the KL-eigenmodes. These matrix-vector products may be performed by the deterministic code by setting a KL-eigenmode as system parameter (indicated by "set parameters" in Fig. 5.2) and executing the matrix-vector multiplication with the system matrices in the deterministic code (indicated by "products with Jacobian" in Fig. 5.2). The solution of linear SPDEs further requires to run the block-preconditioners presented in Section 4.2. These rely on executing the deterministic solver with the mean system parameters (set by the "set parameters" method). Calling the deterministic solver is indicated in Fig. 5.2 by the "solve" method.

If nonlinear SPDEs are solved by the stochastic Galerkin method, realisations of the residual must be computed (see Section 4.4.1). This is indicated in Fig. 5.2 by the "residual" method. Further, the deterministic solver (indicated here by

"solve") needs to be called to precondition the BFGS-solver (see Section 4.4).

These interactions are realised in StoFEL by representing the deterministic code by two abstract classes shown in Fig. 5.3. These classes must be sub-classed to interface to a simulation software for the spatial discretisation.

Figure 5.3: Classes for interfacing to the deterministic code

The Mesh-class: The mesh-class is an abstract class that represents a discretised geometry $R \subset \mathbb{R}^d$. Its *get_points* method returns an array of points that are used to evaluate functions on the spatial domain. Matrix-vector products with the Gram matrix of the spatial ansatz are performed by calling the *mult_gram_matrix* method. As already noted, if the concept of a Gram matrix does not apply, then products with the identity matrix may be performed instead.

Random fields are defined in StoFEL on meshes given by concrete subclasses of the *Mesh*-class. It may be necessary to implement more than one *Mesh*-class for a deterministic code. For example, it may be required to implement one *Mesh*-subclass representing random fields on the domain R and an additional subclass representing random fields on its boundary ∂R.

While this class is called *Mesh*, it is used in a more general manner: it is seen as a vector of values, where the *Mesh*-subclass assigns a special meaning to the vector components. The *Mesh*-class also allows to define random fields without an underlying mesh, for example, vectors of random variables, or nodal vectors for a spectral finite element method.

The PDE-class: A subclass of the abstract class *PDE* must be implemented to interface to a deterministic code. The *PDE* class will be interpreted here as a semi-discretisation of an SPDE, but StoFEL uses it in a more general manner: it may represent any system $\boldsymbol{r}(\boldsymbol{u}(\theta),\theta) = 0$ of stochastic equations; cf. Eq. (3.25).

In addition to the routines shown in Fig. 5.2, the method *get_dof_info()* must be implemented, which returns information about the spatial degrees of freedom. For example, it reports what degrees of freedom correspond to the essential boundary conditions.

A realisation of an SPDE is obtained by assigning values to all relevant parameters of the deterministic code. For this, the *set_params()*-method of the *PDE*-object is called. For example, in the SPDE $-\nabla\cdot(\kappa\nabla u) = f$ the realisations of the

fields κ and f would be specified by calling this method. How the realisations of the fields are specified depends on the deterministic code. For example, if the deterministic code is an FE code, then it may be necessary to specify the values of a field in the operator at the Gauss points of the finite elements (for κ in the above example). It may further be necessary to specify the values of a field on the right hand side for all FE nodes (for f in the above example). Also, fields on the boundary may require special treatment.

For a given nodal vector of solution coefficients of the spatial discretisation and for a given realisation of the SPDE, the *get_resid* computes the coefficients of the residual in the spatial ansatz. Products with the Jacobian are computed by the *mult_jacobian*-method.

The *solve()*-method runs the deterministic solver to obtain the solution of the current SPDE realisation. This routine is used to precondition the linear and the nonlinear solvers. According to Section 4.2 it is not necessary that an exact solution is returned. Convergence of the solvers for the block-system may still be obtained if an inexact block-Jacobi preconditioner is used. For example, if the deterministic solver is an iterative solver, then it may be sufficient to perform only few iterations of the deterministic solver.

How *PDE*-subclasses are implemented and how the deterministic code is coupled is transparent to the other parts of StoFEL. The deterministic code may reside in the same address space as StoFEL and then communications may be performed at small costs. Alternatively, it may be connected to StoFEL by network communication or by other means, resulting in higher communication costs.

Other implementation aspects of the *PDE*-subclass may depend on the costs of communicating with the deterministic code. For example, for linear SPDEs the *mult_jacobian*-routine might be implemented by calling the deterministic code, resulting in high communication demands. Alternatively, the Jacobians (one for each KL-eigenmode of the material parameters) may be retrieved from the deterministic code and stored inside StoFEL, which allows to perform the block-matrix-vector product of Section 4.1 without further communication.

Concrete examples of how deterministic codes were coupled with StoFEL are presented in Section 5.2.

5.1.4 Further Design Aspects

The other modules shown in Fig. 5.1 are addressed only briefly:

The Random Field Discretisation Module: The random field discretisation module comprises classes for the representation and for the discretisation of random fields. It contains classes defining covariance functions, classes defining fields on *Mesh*-objects, a class representing random fields as functions of a finite

number of random variables (as in Section 2.2.4.2), and a class representing random fields as transformations of Gaussian random fields (as in Section 2.2.4.1).

The marginal distributions of transformations of Gaussian random fields are encapsulated in a class hierarchy. New sub-classes may be added to support new marginal distributions for random fields. These sub-classes perform the computation of the second order statistics of the transformed random field and compute its projection onto the polynomial chaos.

For the handling of functions on the spatial domain, the classes in the random field module rely on sub-classes of the *Mesh*-class. This allows to interface transparently to the deterministic code and permits to define random fields on any type of geometry for which a *Mesh*-class is implemented. The random field discretisation module further contains routines that implement the Karhunen–Loève expansion and that obtain a representation in independent random variables.

The Direct Integration Module: Classes for the representation and handling of random variables and for the integration on the resulting probability space are implemented in the Direct Integration module (it is called the Direct Integration and Orthogonal Projection module in Fig. 5.1).

One responsibility of this module is to provide integration formulas for various types of random variables. This module further contains high-dimensional integration routines for the integration of functions on the corresponding probability space, which are constructed from the integration formulas for the individual random variables (see Section B). Monte Carlo methods, full-tensor product quadrature, and Smolyak quadrature formulas based on Gauss formulas and on Clenshaw-Curtis formulas are included in the current implementation.

Additionally, the direct integration module contains routines for the representation of random fields in (generalised) polynomial chaos. To obtain polynomial chaos ansatz spaces, routines are contained which return polynomials that are orthogonal with respect to the probability measure of the basic independent random variables.

Currently, Gaussian and uniformly distributed random variables are implemented. Adding new types of random variables and mixing different types of random variables in one model is simple as the ansatz functions for the stochastic Galerkin method and the integration methods are encapsulated in classes representing the random variables.

Routines for the direct integration of statistics (see Section 3.3) and routines for obtaining a polynomial chaos expansion by orthogonal projection (see Section 3.4.2) are also provided in this module. These routines are versatile and may use any of the high-dimensional integration routines for the SPDE solution.

The Stochastic Galerkin Solver Module: This module contains classes that

perform the stochastic Galerkin-discretisation of an SPDE. These classes rely on a *PDE*-object to obtain the spatial semi-discretisation of the SPDE. As *PDE*-classes may represent any set of equations $\boldsymbol{r}(\boldsymbol{u}(\theta),\theta) = 0$, StoFEL may represent and discretise any stationary system of stochastic equations (but, of course, the solvers discussed in Chapter 4 may not converge for every thinkable system of equations).

Further, the solver module contains classes implementing the solvers for the stochastic Galerkin methods that were discussed in Chapter 4. The classes for the solution of linear SPDEs are arranged in an object-oriented hierarchy, which simplifies the addition of new solvers and which allows to use any solver class as a preconditioner for the Krylov subspace methods. An exception is the parallel solver, which is an external program that was already described in Section 4.3.5.

The classes for the solution of nonlinear SPDEs may use any of the linear solvers as preconditioner. Further, any of the high-dimensional integration routines may be chosen for evaluating the residual.

Apart from these modules, routines for the visualisation of random fields and various utility functions are part of StoFEL.

5.1.5 Implementation

The StoFEL library is a research code. To test new numerical methods quickly it is good to use an interactive programming environment, which allows a rapid prototyping, debugging and the visualisation of intermediate results. The major part of StoFEL was therefore written in MATLAB [106]. This was supplemented by Fortran routines (for coupling to ANSYS) and MEX [105] routines for the communication with the parallel solver. The parallel solver was also implemented using an object-oriented design and was written in C++. The current implementation consists of approximately 30.000 lines of MATLAB code and 10.000 lines of C++ code (including comment lines); see Section 4.3.5 for details on the implementation of the parallel solver.

While more work is required on the solution of large nonlinear SPDEs, the combination of StoFEL with the parallel solver allows to tackle large problems: see Section 4.3, where the parallel solver was applied to SPDEs discretised in many million degrees of freedom.

5.2 Coupling with Existing Solvers

This section presents examples for the coupling of existing simulation software with StoFEL. As discussed in Section 5.1, a deterministic code is coupled with StoFEL by implementing a subclass of the *PDE* class that acts as a wrapper for

the deterministic code. Additionally, it is usually necessary to inherit one or more classes from the *Mesh*-class.

A very simple example of coupling a deterministic code with StoFEL is discussed in Section 5.2.1 and the implementation of this example is shown in Appendix C. The integration of the commercial finite element software ANSYS [4] with StoFEL is described in Section 5.2.2. Finally, Section 5.2.2 discusses briefly the coupling of the commercial finite element software FEMLAB [31] with StoFEL.

5.2.1 A Simple Example

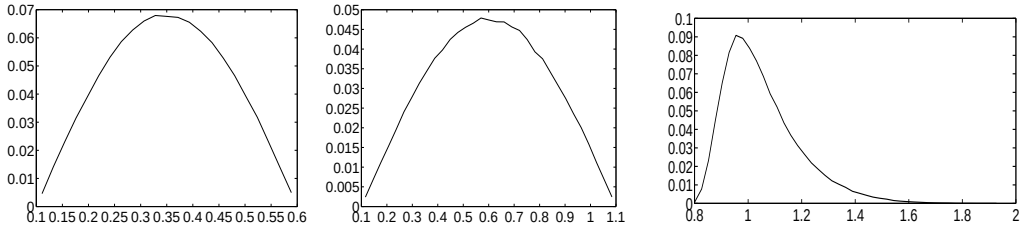
To give an impression of the steps that are required to couple an existing code with StoFEL, the nonlinear spring example of Section 4.4.1 is considered. This is not an SPDE but a simple nonlinear equation with random coefficients

$$a(\boldsymbol{\theta})u(\boldsymbol{\theta}) + b(\boldsymbol{\theta})u(\theta)^3 = f(\boldsymbol{\theta}). \tag{5.1}$$

Here, f and $a, b > 0$ may be constants or random variables and $\boldsymbol{\theta} = (\theta_1, \theta_2)$ is a vector of two mutually independent random variables. The equation is to be solved for the random displacement $u(\boldsymbol{\theta})$ of the spring.

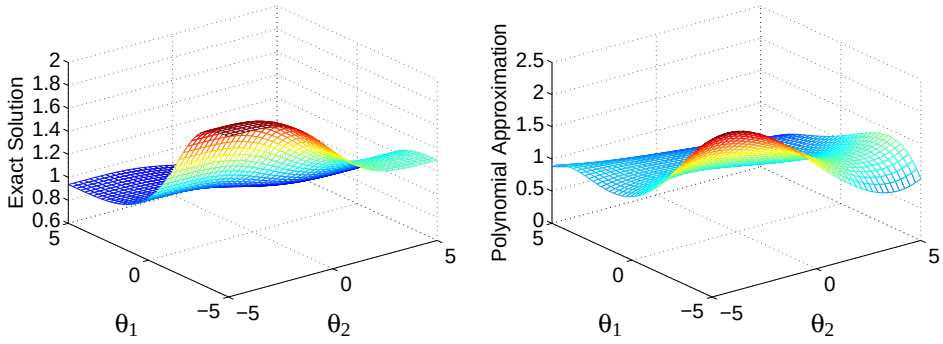
To show how simple it is to implement wrappers for deterministic codes in StoFEL, the complete implementation of Eq. (5.1) in StoFEL is presented and explained in the Appendix C.

Here, solutions of Eq. (5.1) are shown that were computed by the code shown in the Appendix C. The probability densities of $a(\theta_1)$ and $b(\theta_2)$ are shown in Fig. 5.4. They were chosen a bit differently than in Appendix C. By sampling a million times from the solution, its mean $\mu_u = 1.0515$, its standard-deviation $\sigma_u = 0.14$, and its probability density function shown in Fig. 5.4(c) were obtained.



(a) PDF of $a(\theta_1)$ (b) PDF of $b(\theta_2)$ (c) PDF of $u(\theta_1, \theta_2)$

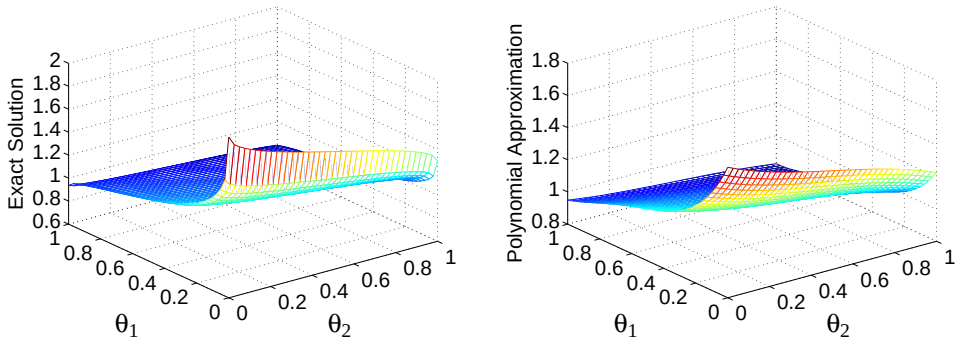
Figure 5.4: Probability densities for Eq. (5.1).



(a) Exact Response Surface

(b) Approximation in Polynomial Chaos of Degree 4

Figure 5.5: Response surface $u(\theta_1, \theta_2)$ for Gaussian θ_1, θ_2.



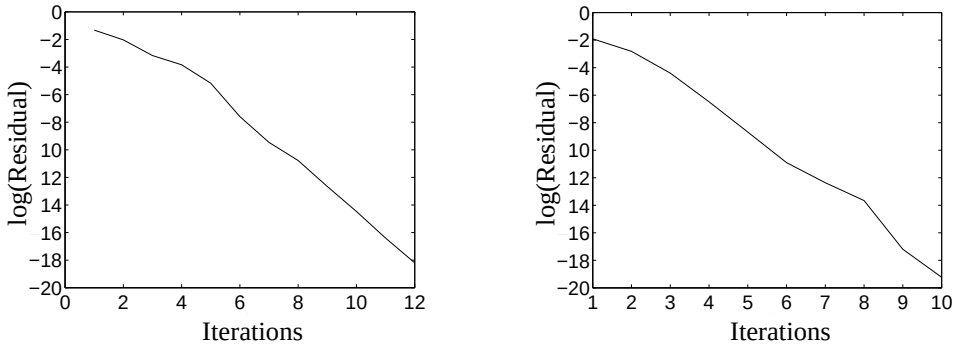
(a) Exact Response Surface

(b) Approximation in Polynomial Chaos of Degree 4

Figure 5.6: Response surface $u(\theta_1, \theta_2)$ for uniformly distributed θ_1, θ_2.

To give an example of generalised polynomial chaos expansions, the same problem was modelled and solved both in Gaussian random variables θ_1, θ_2 and in uniformly distributed random variables θ_1, θ_2.

First, the random variables a and b were modelled as transformations of independent Gaussian random variables θ_1 and θ_2. The resulting nonlinear stochastic equation Eq. (5.1) was solved by the stochastic Galerkin method of Section 3.4.3 in a polynomial chaos ansatz in θ_1 and θ_2 of degree 4. The resulting nonlin-



(a) For Gaussian $\boldsymbol{\theta}$ (b) For uniformly distributed $\boldsymbol{\theta}$

Figure 5.7: Decrease of the residual in the BFGS-solver.

ear system was solved by the BFGS-solver of Section 4.4.2. The mean and the standard-deviation were computed analytically from the response surface. They match well with the above results (with an absolute error of $1.1 \cdot 10^{-4}$ for the mean and an absolute error of $2.2 \cdot 10^{-2}$ for the standard-deviation).

It was mentioned that the polynomial chaos expansion yields a response surface. As this problem depends on only two independent random variables, the response surface may be visualised as a graph: the exact response surface $u(\theta_1, \theta_2)$ plotted as a function of θ_1, θ_2 and its approximation in the polynomial chaos ansatz are shown in Fig. 5.5. When judging their difference, note that the error must be weighted with the Gaussian density $\exp(-(\theta_1^2 + \theta_2^2)/2)$. Taking this into account, they match well (the L_2-error is $2.2 \cdot 10^{-2}$).

The same problem was also modelled in uniformly distributed random variables to show an application of approximations in generalised polynomial chaos. The random variables *a* and *b* were also modelled as transformations of independent uniformly distributed random variables θ_1 and θ_2 (the same statistics as above were chosen for *a* and *b*). Again, the nonlinear stochastic equation was solved by the stochastic Galerkin scheme of Section 3.4.3, but the ansatz was now a generalised polynomial chaos space spanned by multivariate Legendre-polynomials. Again, the nonlinear system was solved by the BFGS-solver and, again, the computed mean and standard-deviation match well with the previously computed results (with an absolute error of $2 \cdot 10^{-4}$ in the mean and an absolute error of $3 \cdot 10^{-2}$ for the standard deviation). As expected, the resulting response surface now looks different than before. Both the exact and the approximated response surface are shown in Fig. 5.5.

It is interesting to note that the BFGS solver shows comparable convergence

behaviour for both cases. This is visualised by the decrease of the residual during the solution of the nonlinear system in Fig. 5.7. In the first case the solver required 12 iterations. In the second case, it required 10 iterations. However, the expansion in Legendre polynomials gave a slightly larger error than the expansion in Hermite polynomials.

This example demonstrates that the same problem may be modelled in different types of independent random variables. Both representations resulted in similar convergence behaviour of the (generalised) polynomial chaos. Nonetheless, this needs not always to be the case and it may be worthwhile to contemplate on the best representation for a given problem. More work in this direction is required; some discussions in this directions can be found in [76, 181, 182, 184, 185].

5.2.2 Coupling with ANSYS

To extend StoFEL's applicability to a large range of problems, the commercial finite element software ANSYS [4] was coupled with StoFEL in [187, 188]. This section describes how the coupling was performed and discusses techniques for visualising the results.

The coupling was implemented for static elastic linear problems, but the extension to other linear stationary problems supported by ANSYS is straightforward. The SPDE supported by the coupling is the elliptic SPDE of static elasticity

$$\begin{aligned} -\nabla\cdot\Big(E(x,\omega)\frac{1}{2}\big(\nabla u(x,\omega)+(\nabla u)^T\big)\Big) &= f_R(x,\omega), \qquad x\in R,\\ u(x,\omega) &= u_0(x,\omega), \qquad x\in\partial R,\\ \big(E(x,\omega)\nabla u(x)\big)\cdot n(x) &= f_N(x,\omega), \qquad x\in\partial R. \end{aligned} \tag{5.2}$$

Here, E is the tensor of elastic moduli, u is the displacement vector, f_R is the body force, and $n(x)$ is the unit normal. Boundary conditions are prescribed in terms of displacements u_0 and in terms of boundary tractions f_N.

The modelling, the meshing, and the definition of loads and boundary conditions are performed in ANSYS, e.g. with its graphical user interface. Following this, the model description is exported to StoFEL, where parameters of the model are defined as random fields. Afterwards, StoFEL discretises the resulting SPDE, where ANSYS is used to obtain the spatial discretisation. In solving the discretised SPDE, the ANSYS solver may be run to precondition the block linear system or to obtain realisations of the solution.

Implementation: The coupling of ANSYS to StoFEL was performed by implementing two subclasses of the *Mesh*-class and a subclass of the *PDE*-subclass.

Hence, any of the discretisation techniques discussed here may be used for solving ANSYS based SPDEs. However, the further exposition concentrates on the stochastic Galerkin-method, where the matrix representation of Section 4.1 is used.

The communication with ANSYS is performed by routines implemented in ANSYS as User Programmable Features (UPFs) [6] and by accessing the binary files that are written when ANSYS performs an analysis [5]; see [187, 188] for details.

Two subclasses of the *Mesh*-class were implemented in StoFEL. After the model has been defined in ANSYS, these *Mesh*-classes retrieve the geometry from ANSYS. Instances of these classes may then be used to define random fields on the spatial domain of the model. For the KL-expansion (see Section 3.1.3), these classes obtain the mass matrix associated with the finite element ansatz from ANSYS.

The current implementation of the *Mesh*-subclasses allows to define random material parameters, random loads, and random Dirichlet conditions. Random Neumann boundary conditions are not supported but might be implemented by an additional *Mesh*-subclass.

For representing ANSYS models in StoFEL, a subclass of the *Pde*-class was implemented (see Section 5.1.3). For the multiplication with the block system matrix and for the computation of the residual it retrieves the stiffness matrices $\boldsymbol{K}_i$ (see Section 4.18) for each KL-eigenmode from ANSYS and stores them in StoFEL. Hence, matrix vector products with the block system matrix and residuals may be computed without further communicating with ANSYS.

Two techniques were implemented for retrieving the stiffness matrices (see [188] for details): one technique retrieves all element stiffness matrices for a unit material parameter from ANSYS. The global stiffness matrices for each KL-eigenmode are then assembled from the element stiffness matrices inside StoFEL without further communication with ANSYS. The other technique calls ANSYS for each KL-eigenmode and retrieves the according stiffness matrices. In all performed experiments, the latter method was significantly slower than the first as it requires more communication.

The preconditioner for the linear solvers was also implemented in two different ways: One implementation of the preconditioner retrieves the mean stiffness matrix from ANSYS and uses MATLAB's linear solver for the preconditioning.

The other implementation sets the mean system parameters in ANSYS and runs the ANSYS solver as preconditioner. This was implemented as a user programmable feature in ANSYS: A user routine was developed that creates the according material models, sets their properties according to the material data, applies the nodal loads as required in the preconditioning stage, solves the problem, and retrieves the corresponding nodal solution. In every iteration of the block linear solver, ANSYS is called for each stochastic degree of freedom to perform the

preconditioning.

In the performed experiments, calling ANSYS was significantly slower than using MATLAB's solver due to the required communication and due to the overhead for setting up the problem in ANSYS. One may expect that using an external FEM software as preconditioner is more advantageous if a finer spatial discretisation is used or if the problem is so complex that specialised solution techniques are required.

As the coupling is performed in a *PDE*-subclass, all discretisation techniques discussed previously may be applied to solve ANSYS based SPDEs. Further, all discussed solvers, including the parallel solver, may be used to solve the block system of equations resulting from the stochastic Galerkin discretisation.

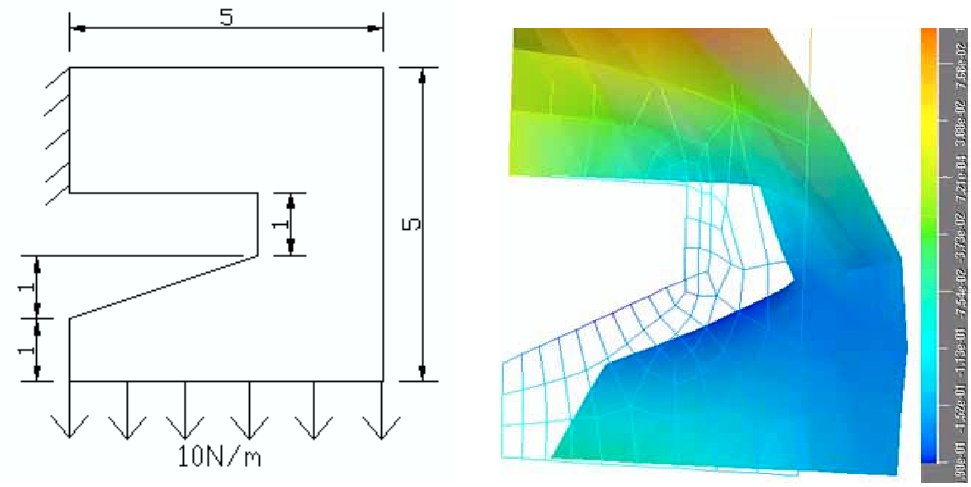
Post-processing: Random fields are functions on a high-dimensional space. Hence, special visualisation techniques are required; see the review [80] for an overview.

Usually, random fields are visualised by displaying statistics. Techniques for the visualisation of the second order statistics and of the cumulative density functions of a random field were implemented in [187, 188]. These techniques may be used to visualise any random field expanded in polynomial chaos, but they are used here only for results that were computed by the coupling of StoFEL and ANSYS.

Statistics of the solution are visualised here using the software AVS (Application Visualisation System) [2]. The mesh of the spatial domain is converted to unstructured cell data (UCD), and all univariate statistics are treated as nodal data. Such statistics are, for example, the mean, the variance, or the covariance between a fixed node and all other nodes. These statistics are represented as unstructured cell data, which allows to use standard techniques for their visualisation.

Further, techniques for the visualisation of the cumulative distribution functions (CDFs) of random fields on two-dimensional domains $R \subset \mathbb{R}^2$ were implemented. The CDF at a point $x \in R$ is defined as $F_{u(x)}(\hat{u}) := P\{u(x) \leq \hat{u}\}$. It is a function $F_u \colon R \times \mathbb{R} \to \mathbb{R}$ with $(x, \hat{u}) \mapsto F_{u(x)}(\hat{u})$ and may hence be visualised as volume data in $R \times \mathbb{R}$. Note that the marginal probability density function (PDF) of a random field is also a function $f_u \colon R \times \mathbb{R} \to \mathbb{R}$. Hence, the same techniques used here to visualise the CDF may be used to visualise the PDF of a random field.

The visualisation of the CDF as volume data requires to convert the information in the CDF into a three-dimensional UCD structure. This UCD structure is constructed of several layers (see Fig. 5.10). The middle layer is given by the mesh corresponding to the mean response. It consists of the FE mesh in the points $(x_i, \mathbf{E}(u(x_i))), i = 1, \ldots, n$. Here, $x_i \in R$ are the nodal positions of the finite elements and $\mathbf{E}(u(x_i))$ are the mean values of the solution in this nodal value.



(a) Geometry of Plate

(b) Mean horizontal displacement (enlarged by 7)

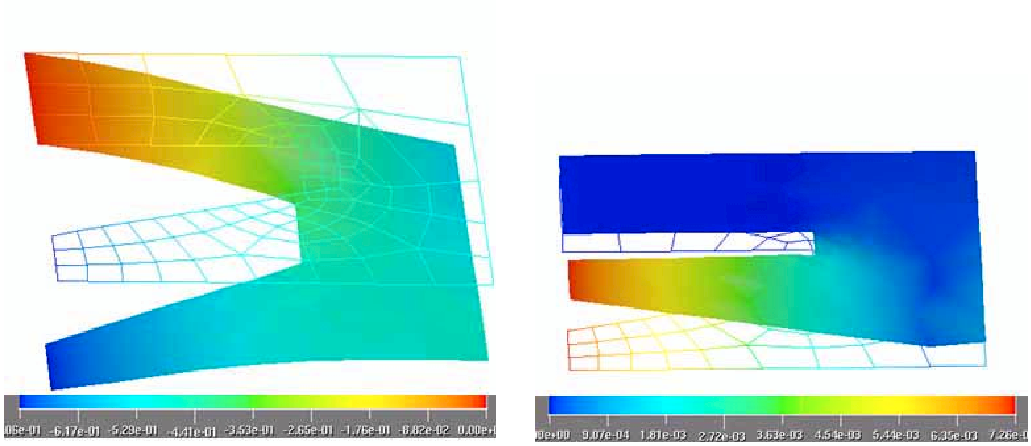
Figure 5.8: Geometry and mean horizontal Displacement

In every node of this mean surface an axis in the z-direction is constructed (see Fig. 5.10), which is cut into two halves by the mean surface. This axis extends from the mean minus a multiple of the standard deviation to the mean plus a multiple of the standard deviation. Each of these axes is divided into equidistant points and copies of the mean surface are constructed in these points. The resulting surfaces are then connected to form a three-dimensional UCD structure (see the grids in Fig. 5.10). The CDF (or the PDF) is represented as nodal data on this UCD structure and standard volume visualisation techniques are used to display the CDF (or the PDF) of the random field.

Examples: As examples, a plate and a three-dimensional elastic structure with random Young's modulus were solved. The following results were published in [187, 188].

The structure, the constraints, and the loads of the first example are shown in Fig. 5.8(a). The Young's modulus of this plate is defined as the lognormally distributed random field $E(x,\omega) = \exp(7.5959 + 0.0998\gamma(x,\omega))$, where γ is a centred Gaussian random field with $\mathrm{cov}_\gamma(x,y) = \exp(-\|x-y\|^2/0.2^2), x,y \in R$.

The resulting SPDE Eq. (5.2) was solved by StoFEL coupled with ANSYS, where the three-dimensional polynomial chaos of degree three was used for the stochastic discretisation. The mean of the horizontal component of the displacement is shown in Fig. 5.8(b). The mean vertical displacement and its standard



(a) Mean of vertical displacement (enlarged by 2)

(b) Standard Deviation of vertical displacement (enlarged by 100)

Figure 5.9: Examples computed by ANSYS

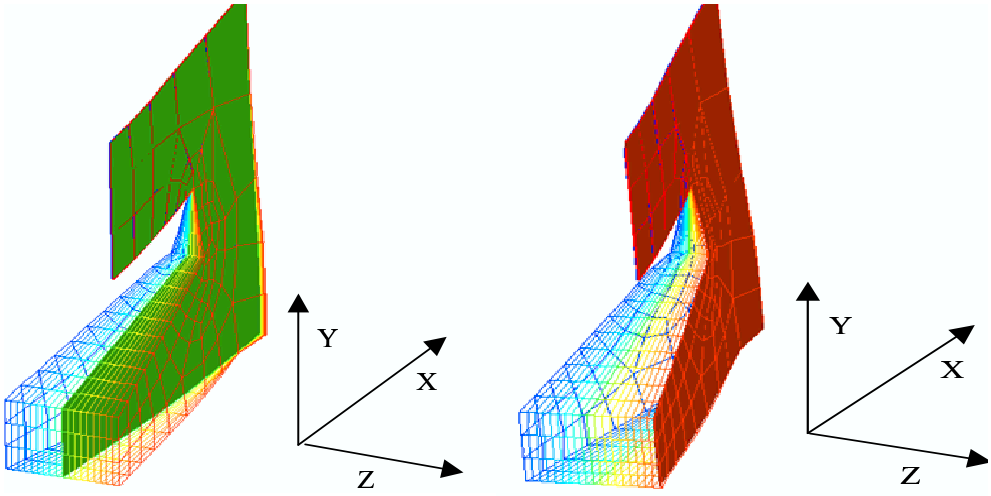
deviation are shown in Fig. 5.9.

The cumulative density function $F_{u_x(x)}(\hat{u})$ of the horizontal component is visualised in Fig. 5.10 by showing the iso-surfaces of all $(x,\hat{u}), x \in R, \hat{u} \in \mathbb{R}$ with $F_{u_x(x)}(\hat{u}) = 0.5$ (the median) and with $F_{u_x(x)}(\hat{u}) = 0.99$ (the 99% quantile). The z-coordinates in Fig. 5.10 correspond to the possible values of $u_x(x)$ and the grid extends from the mean minus three times the standard deviation up to the mean plus three times the standard deviation.

The result of a three-dimensional stochastic structural analysis is shown in Fig. 5.11. The geometry is shown in Fig. 5.11(a). Young's modulus was chosen as $E(x,\omega) = \exp(14.489 + 0.198\gamma(x,\omega))$, where γ is a standard Gaussian random field with covariance function $\mathrm{cov}_\gamma(x_1,x_2) = \exp(-\|x_1 - x_2\|/0.5^2)$. The displacements are now volume data. They are visualised in Fig. 5.11(b) and in Fig. 5.11(c) by showing them as surfaces over two-dimensional intersections with the structure. The standard-deviation is visualised as a graph over a two-dimensional section and as an iso-surface in Fig. 5.11(c).

5.2.3 Coupling with FEMLAB

Additionally to the solvers discussed above, the MATLAB [106] based commercial finite element program FEMLAB [31] was coupled with StoFEL. FEMLAB is a general purpose tool for the modelling and for the simulation of physical



(a) Iso-surface $F_{u(x)}(\hat{u}) = 50\%$ (Median)

(b) Iso-surface $F_{u(x)}(\hat{u}) = 99\%$, (99% quantile)

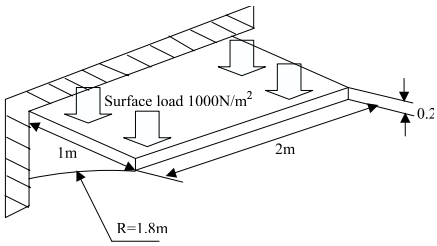
Figure 5.10: Examples computed by ANSYS

systems described by arbitrary partial differential equations. Models may be constructed interactively in the FEMLAB graphical user interface. Afterwards, the parameters of these models may be defined as random fields in StoFEL.

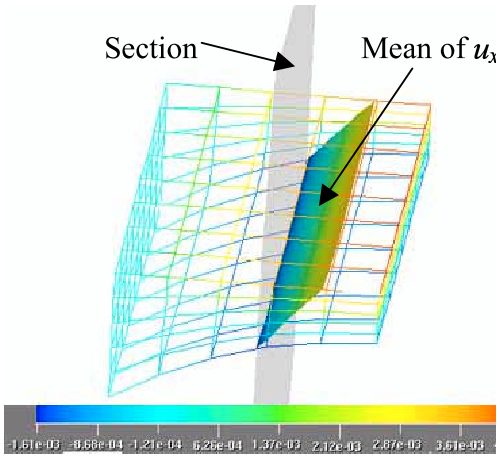
The interface between StoFEL to FEMLAB allows to define and to discretise SPDEs on one-, two-, or three-dimensional spatial domains $R \subset \mathbb{R}^d, d = 1,\ldots,3$, that correspond to all types of stationary linear or nonlinear PDEs supported by FEMLAB.

A significant advantage compared to the coupling with ANSYS is that both FEMLAB and StoFEL are MATLAB [106] based. Hence, the communication overhead is negligible. This makes the solution of nonlinear SPDEs feasible, which requires much communication with the deterministic code.

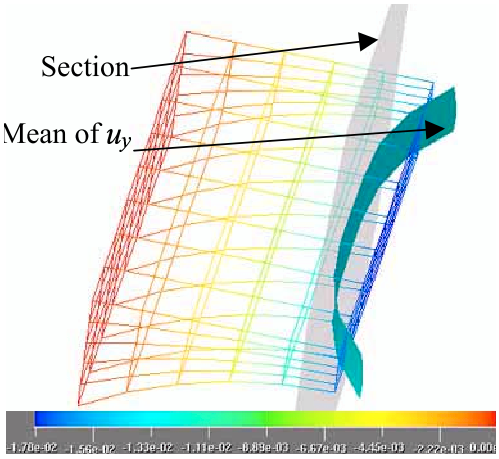
Most examples of this thesis were solved using the FEMLAB interface. Therefore, no additional examples are shown here. An example of a linear FEMLAB based SPDE was shown in Section 2.1. Examples for nonlinear SPDEs discretised by FEMLAB were shown in Section 3.3.4 and in Section 4.4.3. Also, the models solved by the parallel solver Section 4.3 were FEMLAB based.



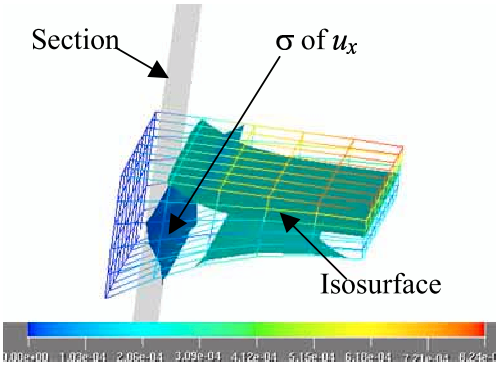
(a) Geometry and Loads of three-dimensional elastic problem



(b) Mean of horizontal displacement on grey section



(c) Mean of vertical displacement on grey section



(d) Standard Deviation of mean displacement shown as Iso-surface

Figure 5.11: Three-Dimensional Examples computed by ANSYS

Chapter 6

Conclusions

The present work develops numerical techniques for the solution of systems with stochastic uncertainties. These numerical techniques were implemented in a general-purpose software that utilises existing simulation codes (called here the deterministic code or the deterministic solver) for the spatial discretisation.

The findings and contributions of the present thesis are summarised below together with recommendations for further research.

SPDE Theory: Existence and uniqueness proofs for solutions of linear elliptic SPDEs and of nonlinear, strictly monotone, hemicontinuous, coercive SPDEs were given in Section 2.3.

An open problem in the theory of SPDEs is the regularity of the solution. As discussed in Section 3.4.4, there are simple results for SPDEs depending on only a finite number of random variables, but these cannot be generalised to general probability spaces as an infinite-dimensional calculus is required there. It is not clear how the stochastic regularity (e.g. expressed in Hida- or Kondratiev-spaces of stochastic distributions or test functions) of the solution depends on the regularity of the random fields.

Random Field Discretisation: A representation of the SPDE in a countable number of mutually independent random variables $\theta_1, \theta_2, \ldots$ was obtained in Section 3.1.1 by Karhunen–Loève expansions. One goal was to perform the spatial discretisation by existing simulation software used in a black-box fashion. Hence, a technique was developed for discretising the KL-eigenproblem by existing finite element software (see Section 3.1.3). Explicit estimates for the introduced error were derived (Section 3.1.4). In principle, these allow to compute à posteriori errors for the truncated KL-expansion, but this may be too costly in practice. The specification of à priori estimates for the error of the truncated KL-expansion is an open problem.

Direct Integration Methods: These methods approximate the SPDE in a finite number of independent random variables and compute statistics by numerically integrating functionals of the solution of this perturbed SPDE. Stability with respect to these perturbations was shown in Section 3.3.1 for the types of random field approximations used here.

Monte Carlo integration and Smolyak quadrature were used for the integration. Other techniques for high-dimensional integration were reviewed in Section 3.3.3 and in Appendix B. In the experiments performed in Section 3.3.4 Smolyak quadrature was superior to Monte Carlo methods for the direct computation of statistics.

Series Expansion Methods: The series expansion methods discussed in Section 3.4 expand the solution in an ansatz of tensor products of spatial times stochastic functions. Here, orthogonal projections (Section 3.4.2) and Galerkin methods (Section 3.4.3) were used for the stochastic discretisation.

It was remarked that, in contrast to the direct integration methods, the Galerkin methods do not require to perturb the SPDE by approximating it in a finite number of random variables. Section 4.1 showed how this may be done in practice.

Some steps towards obtaining à priori convergence rates were reviewed (Section 3.4.4). However, to apply them, a regularity theory for SPDEs is required, which has not yet been devised.

High-Dimensional Problems: The stochastic dimension is the number of the independent random variables used in discretising the SPDE. It is determined by properties of the random fields used in the SPDE, like the smoothness and the correlation length of their covariance functions (see Section 3.1.1).

The stochastic dimensions influence both the direct integration methods and the series expansions methods. It is the high dimension and not per se the existence of stochastic uncertainties that makes the numerical solution of SPDEs difficult. It is therefore surprising that high-dimensional problems are seldom tackled when stochastic discretisation methods are presented or compared in the literature (cf. the review [80]).

The influence of the dimensions in the context of direct integration methods was discussed in Section 3.3 and in the Appendix B. An abundant literature on high-dimensional integration (see Appendix B) is available and their findings are often directly applicable to the solution of SDPEs. Surprisingly, this research area has so far hardly been taken into account in stochastic mechanics.

In the context of series expansion methods, the stochastic dimensions affect the choice of the stochastic ansatz: In principle, the stochastic Galerkin methods permit to use any finite-dimensional space of admissible functions as ansatz (see Section 3.4 and the review [80] for details). However, the high dimensions compli-

cate this in practice. For example, it was discussed in Section 3.4 that piecewise polynomials on regular meshes are not tractable, as they result in a number of equations growing exponentially with stochastic dimensions.

This thesis, as many previous works, uses spaces of global polynomials with a given maximum total degree (so-called generalised polynomial chaos spaces). Here, the ansatz size grows only polynomially with the stochastic dimensions. The size of the discretisation still grows fast with the stochastic dimensions, hence these ansatz spaces are feasible for the discretisation of SPDEs in low to medium dimensions. For example, stochastic Galerkin methods were applied here to polynomial chaos expansions in up to 64 stochastic dimensions.

Remedies for the resulting high numerical effort may be adaptive techniques (see below) and alternative ansatz spaces. For example, sparse tensor product ansatz spaces [61, 159] merit attention in stochastic mechanics.

Choice of the Discretisation Method: How to choose an appropriate technique for the discretisation of a given SPDE is not sufficiently understood. As well as one may construct experiments where stochastic Galerkin methods are superior to Monte Carlo and other integration techniques (small stochastic dimensions, high variance), one may also construct experiments where Monte Carlo techniques are favourable (high stochastic dimensions, low variance).

Comparisons found in the literature are often unfair (cf. the review [80]): Often, series expansion techniques in small stochastic dimensions are compared to Monte Carlo methods and not to numerical integration methods appropriate for the small stochastic dimensions. Moreover, comparisons between series expansion techniques (e.g. stochastic Galerkin methods or perturbation methods) and direct integration techniques (e.g. Monte Carlo) must always be considered with care as they are only meaningful with respect to the stochastic dimensions used and with respect to the statistics being computed.

Section 3.3.3 and Appendix B discussed how the efficiency of integration techniques depends on the stochastic dimensions and on properties of the integrand. Similarly, the efficiency of series expansion methods was seen in Section 3.4.4 to depend on approximation properties of the ansatz spaces with respect to the stochastic regularity of the response.

The following qualitative guidelines may be given (cf. the discussions in Section 3.3.3, Section 3.4.4 and Appendix B): If the functional (the statistics) of interest has an integrand with large variance, then Monte Carlo methods perform badly. If the integrand is smooth, then polynomial expansions and Smolyak quadrature methods based on Gauss or Clenshaw-Curtis rules may be well-suited. If a high accuracy is required, then Monte Carlo methods are usually a bad choice. If small failure probabilities are to be computed, then specialised integration techniques should be used, like the first or second order reliability methods.

The method of choice is further determined by properties of the SPDE: the input random fields determine the required stochastic dimensions. If the stochastic dimensions are large, then it is hard to beat Monte Carlo methods. If the stochastic dimensions are small to medium, then (interpolatory) quadrature techniques or polynomial series expansion techniques may be good choices. For linear SPDEs, the stochastic Galerkin solvers presented in Chapter 4 may be highly effective. Also, the solvers for nonlinear SPDEs presented in Section 4.4 were more effective than the direct integration methods in the experiments performed here.

The above discussion is of a qualitative nature. More investigations are required to give explicit guidelines on how to choose an appropriate discretisation method for computing statistics of SPDEs.

Galerkin Solvers for Linear SPDEs: This work concentrates on stochastic Galerkin methods. These yield large systems of coupled block equations. Solvers for these equations are the focus of this thesis.

For linear SPDEs, these block equations are highly structured. An efficient representation for the resulting block matrix was obtained in Section 4.1.1 by expanding the stochastic operator in polynomial chaos. The expansion was shown to be a finite sum, which allows to work with unperturbed SPDEs in practice and which yields a practical criterion for choosing the operator expansion.

A more advantageous representation of the block matrix was developed in Section 4.1.2: the Karhunen–Loève expansion of the (non-Gaussian) material parameters in the operator was computed and the thus obtained uncorrelated random variables were expanded in polynomial chaos. The resulting approximation of the operator allows to compute matrix-vector products with the system block matrix at small costs. It is well-suited for a parallelisation of the block matrix-vector product and it allows to couple to the deterministic codes with less computational effort than the first representation.

Both representations allow the efficient execution of the block matrix-vector product, hence the linear block system was solved here by iterative methods. Block versions of the classical iterative methods, preconditioned Krylov subspace methods, and multi-level methods were developed, where the preconditioning was performed by repeatedly calling the deterministic solver in a black-box fashion.

Parallel Solver for Linear SPDEs: Due to the size of the resulting system of equations, a parallel solver was implemented for the solution of linear SPDEs discretised by stochastic Galerkin methods. The parallelisation was performed in a configurable manner, exploiting various hierarchies of parallelism.

The experiments in Section 4.3.6 show that the parallel solvers allows to discretise and to solve linear SPDEs in many millions of degrees of freedom. Depending on the type of the parallelisation, good to almost perfect speedups were

obtained.

An aspect of the parallel solvers that requires more work is the coupling with existing simulation software. The parallel solver already now allows the deterministic code to be a parallel program: Processors groups running the deterministic code are the smallest building blocks for the coarser levels of parallelism. However, the deterministic solver is currently a simple self-written parallel conjugate gradient method (see Section 4.3.5). More work is required to couple the parallel solvers transparently with specialised parallel codes.

Galerkin Solvers for Nonlinear SPDEs: Numerical methods for nonlinear SPDEs were developed in Section 4.4, where the discretisation was performed by stochastic Galerkin methods and by orthogonal projection techniques. Again, the deterministic code was used in a black-box fashion for the spatial discretisation.

The systems of nonlinear block equations resulting in the stochastic Galerkin methods were solved by approximate Newton methods and by quasi-Newton methods. Both nonlinear solvers rely on the solvers for the linear SPDEs: the approximate Newton method in each iteration runs a solver for linear SPDEs to obtain the next correction. The quasi-Newton method avoids this nested iteration and is preconditioned by one or more iterations of a block-diagonal preconditioner.

A problem in solving nonlinear SPDEs by Galerkin methods was the evaluation of the residual. It was found (see Section 4.4.1) that Monte Carlo methods may be badly suited for this, because the oscillating nature of the stochastic ansatz causes the integrands of the residual to have a high variance. Hence, alternative quadrature rules were used and high-dimensional quadrature methods were shown to be an efficient alternative.

Adaptivity and Sensitivity: It was discussed in Section 4.5 that several techniques used in the literature for the adaptive refinement or for the computation of sensitivities are based on employing derivatives with respect to the basic independent random variables. However, an infinite-dimensional calculus is required in general stochastic spaces. It is therefore not clear whether sensitivities computed by these techniques are stable approximations of sensitivities of the original SPDE.

The adaptive method implemented here (Section 4.5) does not employ such derivatives: It uses a goal-oriented approach based on solving a problem dual to the original one. The solution of the dual problem may be interpreted as a sensitivity, but this was not investigated here further.

Adaptivity was implemented here only for linear SPDEs and only by refining the stochastic ansatz, but by generalising the techniques used for PDES [101, 140, 145], the same ideas might also be applied to nonlinear SPDEs, to nonlinear

functionals, or to implement space-stochastic adaptivity.

General Purpose Software Framework: The numerical techniques developed here were implemented in the general purpose software for stochastic finite elements StoFEL ("Stochastic Finite Element Library"); see Chapter 5.

StoFEL permits to introduce stochastic uncertainties into existing simulation codes without changing these. It allows much versatility in defining random fields and SPDEs: Random fields may be defined on any type of geometry supported by the deterministic code. An arbitrary number of random fields may be used in the operator, in the right hand side, or in the boundary conditions; discretised random fields are represented as functions of independent random variables; different types of random variables (e.g. Gaussian and uniformly distributed random variables) may be mixed in one model. The object-oriented design simplifies the addition of new types of random fields and random variables.

Its modular, object-oriented architecture allows to apply StoFEL to a wide range of applications. The spatial semi-discretisation of the SPDE is encapsulated in a class, which may also represent systems of stochastic equations that do not stem from SPDEs. Examples for the coupling with the commercial finite element codes FEMLAB [31] and ANSYS [4] demonstrate the generality of this approach (see Section 5.2). The applicability of StoFEL to large problems was demonstrated by solving systems with many millions of unknowns (see Section 4.3.6).

Appendix A

Stochastic Analysis

Some basics of probability theory (see e.g. [14, 63, 133]) and of stochastic analysis are presented here.

A.1 Basics of Probability Theory

In the following, $(\Omega, \mathcal{B}, P)$ denotes a probability space, where Ω is the set of elementary events, $\mathcal{B}$ is the σ-algebra of events and P is the probability measure. The symbol ω always specifies an elementary event $\omega \in \Omega$.

Random variables (RVs) are measurable functions $\kappa\colon \Omega \to V$, where V is a measure space and they are written here in Greek letters. If $V = \mathbb{R}^d$, then $\boldsymbol{\kappa}$ is a *random vector* and this is emphasised by bold Greek letters. The σ-algebra generated by a set of random-variables $\{\kappa_i\}_{i\in\mathcal{I}}$ with an index set $\mathcal{I}$ is denoted $\Sigma(\{\kappa_i\}_{i\in\mathcal{I}})$.

Every random variable κ with values in V induces a probability measure on V that is called P_κ. The distribution function of a real valued random variable κ is called $F_\kappa(k) = P_\kappa(-\infty, k) = P\{\kappa < k\}$ and—if it exist—its probability density is denoted by $p_\kappa(k) = \frac{dF_\kappa(k)}{dk}$.

Random variables are often characterised by their statistics given by functionals that are defined as

$$\mathbf{E}\left(g(\kappa)\right) = \int_\Omega g(\kappa(\omega))\, dP(\omega) = \int_{\mathbb{R}} g(\kappa)\, dF_\kappa(\kappa), \tag{A.1}$$

where g is an appropriate function. Some important statistics are the mean $\mu_\kappa = \mathbf{E}\left(\kappa\right)$, the variance $\mathrm{var}_\kappa = \mathbf{E}\left(\kappa^2\right) - \mu_\kappa^2$, and the standard deviation $\sigma_\kappa = \sqrt{\mathrm{var}_\kappa}$. The probability that κ takes values in a P-measurable set $B \subset V$ may also be written in this way as $P\{\kappa \in B\} = \mathbf{E}\left(\chi_B(\kappa)\right)$, where χ_B is the characteristic function of B.

The covariance is a bivariate statistics of two random variables κ_1 and κ_2, $\mathrm{cov}(\kappa_1,\kappa_2) = \mathbf{E}\left((\kappa_1-\mu_{\kappa_1})(\kappa_2-\mu_{\kappa_2})\right)$. In general, multivariate statistics are expectations of functions $g(\kappa_1,\ldots,\kappa_m)$ in more than one random variable and may be written for real-valued $\kappa_1,\ldots,\kappa_m$ as

$$\mathbf{E}\left(g(\kappa_1,\ldots,\kappa_m)\right) = \int_{\mathbb{R}}\cdots\int_{\mathbb{R}} g(k_1,\ldots,k_m)\, dF_{\kappa_1,\ldots,\kappa_m}(k_1,\ldots,k_m), \tag{A.2}$$

where $F_{\kappa_1,\ldots,\kappa_m}$ is the joint distribution function of $\kappa_1,\ldots,\kappa_m$.

A connection to numerical procedures is made clear in Section 3.3: When stochastic problems are discretised, one usually starts with an abstract probability space $(\Omega,\mathcal{B},P)$. Then the problem is represented in a finite number of independent random variables $\boldsymbol{\theta}(\omega) = (\theta_1(\omega),\ldots,\theta_m(\omega))^T$. Practical computations may then be performed on the probability space induced by $P_{\boldsymbol{\theta}}$ on the range of the random vector $\boldsymbol{\theta}$. If the $\theta_1,\ldots,\theta_m$ are independent, then Fubini's lemma may be used to compute a multivariate statistics of the type given in Eq. (A.2) as

$$\mathbf{E}\left(g(\theta_1,\ldots,\theta_m)\right) = \int_{\mathbb{R}}\cdots\int_{\mathbb{R}} g(\theta_1,\ldots,\theta_m)\, dF_{\theta_1}(\theta_1)\cdots dF_{\theta_m}(\theta_m). \tag{A.3}$$

Such "coordinate systems" of independent RVs will usually be given by vectors of independent Gaussian RVs. The reasons for this are that two Gaussian RVs θ_1,θ_2 are independent if they are uncorrelated, i.e. if $\mathrm{cov}(\theta_1,\theta_2)=0$, and that their linear combinations are also Gaussian. Hence, Gaussian RVs may be transformed to independent RVs by linear algebra.

A Gaussian RV γ with mean μ and standard deviation σ will be denoted by $\mathcal{N}(\mu,\sigma^2)$. Its probability density function is

$$f_\gamma(x) = \frac{1}{(2\pi)^{1/2}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{A.4}$$

A centred Gaussian random variable with unit variance, $\mathcal{N}(0,1)$, is called *standard Gaussian*. The probability distribution function of a standard Gaussian random variable will be called $\mathrm{erf}(x) := F_{\mathcal{N}(0,1)}(x)$.

Stochastic properties will often be specified as nonlinear transformations of Gaussian random variables. For this, the well-known fact will be employed that a standard Gaussian random variable is mapped to a random variable with distribution function F_κ by the transformation $F_\kappa^{-1}(\mathrm{erf}(\mathcal{N}(0,1)))$ [e.g. 133].

A.2 Spaces of Random Variables

Some basics of stochastic analysis are needed; for introductions see e.g. [75, 100].

A.2.1 Gaussian Banach and Hilbert Spaces

The L^p norms for random variables are defined as usual, $\|\kappa\|_p = \mathbf{E}(\kappa^p)^{1/p}$ for $0 < p < \infty$ and $\|\kappa\|_\infty = \operatorname{ess\,sup}|\kappa|$. The space $L^p = L^p(\Omega, \mathcal{B}, P)$ is the space of all random variables on $(\Omega, \mathcal{B}, P)$ that satisfy $\|\kappa\|_p < \infty$. Just as in the deterministic case, L^2 is a Hilbert space, and L^p is a Banach space for $1 \leq p \leq \infty$. For $1 \leq p < \infty$, the dual is $(L^p)' = L^q$ with $q^{-1} + p^{-1} = 1$. Furthermore, L^r is a dense subset of L^p whenever $0 < p \leq r \leq \infty$.

For centred variables $\gamma_1, \gamma_2 \in L^2$, the expression $(\gamma_1, \gamma_2)_{L^2} := \operatorname{cov}(\gamma_1, \gamma_2)$ defines a scalar product with norm $\|\gamma\|_2^2 := \operatorname{var}_\gamma$. A *Gaussian Hilbert space* [e.g. 75] Θ is a subspace Θ of $L^2(\Omega, \mathcal{B}, P)$ that only contains centred Gaussian random variables and that is complete when equipped with this scalar product. Note that the σ-algebra $\Sigma(\Theta)$ may be smaller than $\mathcal{B}$; this is the usual case when stochastic quantities are approximated in a finite number of independent random variables.

An important example of a Gaussian Hilbert space is the linear span of m independent standard Gaussian random variables $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$. This space may be identified with $(\mathbb{R}^m, \mathcal{B}_m, P_{\boldsymbol{\theta}})$, where $P_{\boldsymbol{\theta}}$ is the Gauss measure, $dP_{\boldsymbol{\theta}}(x) = (2\pi)^{-m/2} \exp(-|x|^2/2)\, dx$ and where $\mathcal{B}_m$ is the Borel σ-algebra on $\mathbb{R}^m$.

A.2.2 Measures on Topological Vector Spaces

Random fields are discussed in Section 2.2. They are collections of random variables $\kappa(x)$ indexed by $x \in R \subset \mathbb{R}^d$, but they may also be interpreted as RVs κ with values in a function space V. Let us hence discuss how a probability measure may be constructed on a topological vector space V, which will be assumed to be Banach or locally convex. For clearness, Gaussian measures [see 75, example 1.13] are discussed first and general measures afterwards.

(a) A Borel probability measure P on V (with the Borel σ-algebra $\mathcal{B}$) is said to be Gaussian, if each continuous linear functional $v' \in V'$ regarded as a random variable on $(V, \mathcal{B}, P)$ is Gaussian. If each $v' \in V'$ is centred Gaussian, then the completion of $V' \subseteq L^2(V, \mathcal{B}, P)$ is a Gaussian Hilbert space.

(b) A random variable κ with values in V is Gaussian if $\omega \mapsto \langle v', \kappa(\omega) \rangle$ is a Gaussian random variable for each $v' \in V'$, where $\langle \cdot, \cdot \rangle$ is the duality paring on $V' \times V$. If κ is centred, then the set of all $\langle v', \kappa(\cdot) \rangle, v' \in V'$, is a linear space isomorphic to the space V' considered in (a).

In general, measures are specified on the dual V' of a given topological vector space V by their Fourier transform: A V'-valued random variable κ is specified by a generalisation of the characteristic function for real valued variables [e.g. 133],

the *characteristic functional*. It is defined for $v \in V$ as

$$\Phi_\kappa(v) := \mathbf{E}\left(e^{i\langle\kappa(\cdot),v\rangle}\right) = \int_{V'} \exp\left(i\langle v', v\rangle\right) dP_\kappa(v'). \tag{A.5}$$

For a given functional Φ_κ on a nuclear space V, the following theorem states conditions for the existence of a measure P_κ on V':

Theorem A.1: *(*Bochner-Minlos, see [44, Theorem 2, p. 350]*) Any continuous, positive definite functional Φ_κ on a nuclear space V with $\Phi_\kappa(0) = 1$ is the Fourier transform Eq. (A.5) of a countably additive positive normalised measure P_κ on V'.*

For example, the functional $\Phi_\gamma(v) := e^{-1/2\|v\|_{L^2}^2}$ on the space of rapidly decreasing functions $V = S(\mathbb{R}^d)$ satisfies the conditions of the Bochner-Minlos theorem. It thus defines a Gaussian probability measure P_γ on the space of tempered distributions $V' = (S(\mathbb{R}^d))'$ with the Borel σ-algebra $\mathcal{B}$ of $S'(\mathbb{R}^d)$ equipped with the weak-$*$-topology. The measure P_γ is called the *d-parameter white noise measure*, and the corresponding V'-valued random variable γ is called the *d-parameter white noise process* [70].

The requirement in Theorem A.1 that V is nuclear is important—for example, it is not possible to define a Gaussian measure on an infinite-dimensional Hilbert space [e.g. 25, Chapter VII]—a Gaussian measure may instead be found on a larger topological vector space into which the Hilbert space is densely embedded [e.g. 75, Example 1.25].

For more examples of spaces with Gaussian probability measure, see [70].

A.2.3 Polynomial Chaos

The *polynomial chaos* is also called the *Wiener polynomial chaos*, the *Wiener chaos*, or the *Wiener-Itô Chaos*. The name may be misleading: the polynomial chaos is a space of orthogonal polynomials and the name was termed by its inventor Norbert Wiener [180] long before the modern meaning of the word "chaos" was established.

In the following, let $\Theta \subseteq L^2(\Omega, \mathcal{B}, P)$ be a separable Gaussian Hilbert space. The space of all multivariate polynomials of degree k is denoted by

$$\mathcal{P}_k(\Theta) := \{p(\theta_1, \ldots, \theta_m) \mid p \text{ is polynomial of degree } k; \theta_1, \ldots, \theta_m \in \Theta, m < \infty\}. \tag{A.6}$$

The space of all polynomials will be called $\mathcal{P}(\Theta) := \cup_{k=0}^{\infty} \mathcal{P}_k(\Theta)$. Denote by $\bar{\mathcal{P}}_k$ the closure with respect to L^2 and define

$$\mathcal{H}_{=0} := \bar{\mathcal{P}}_0, \qquad \text{(the space of constants)} \tag{A.7}$$

$$\mathcal{H}_{=k} := \bar{\mathcal{P}}_k \ominus \bar{\mathcal{P}}_{k-1}, \qquad k \in \mathbb{N}. \tag{A.8}$$

$h_0(x)$	$=$	1	$h_3(x)$	$=$	x^3-3x
$h_1(x)$	$=$	x	$h_4(x)$	$=$	x^4-6x^2+3
$h_2(x)$	$=$	x^2-1	$h_5(x)$	$=$	x^5-10x^3+15x

Table A.1: Hermite polynomials (unnormalised)

The vector space $\mathcal{H}_{=k}$ is called the *homogeneous chaos of order k* and the vector space $\mathcal{H}_{\leq k} := \cup_{l=0}^{k} \mathcal{H}_{=k}$ is called the *polynomial chaos of order k*.

The space of polynomials $\mathcal{P}(\Theta)$ is dense in $L^p(\Omega, \sigma(\Theta), P)$ for $0 < p < \infty$ [e.g. 75, Theorem 2.11], and the orthogonal decomposition [e.g. 75, Theorem 2.6]

$$L^2(\Omega, \Sigma(\Theta), P) = \bigoplus_{k=0}^{\infty} \mathcal{H}_{=k} \tag{A.9}$$

is called the *polynomial chaos decomposition* or the *Wiener chaos decomposition* of $L^2(\Omega, \Sigma(\Theta), P)$ [180].

An orthogonal basis of the polynomial chaos may explicitly be constructed by multivariate Hermite polynomials. These are tensor-products of (univariate) Hermite polynomials h_k, where $k \in \mathbb{N}_0$ specifies their degree; see Table A.1. Note that the h_k are orthogonal with respect to the Gaussian measure.

The construction of the polynomial chaos uses *multi-indices*. These are sequences $\alpha = (\alpha_i)_{i \in \mathbb{N}}$ of non-negative integers with only finitely many non-zero elements. The set of all multi-indices will be denoted by

$$(\mathbb{N}_0)_c^{\mathbb{N}} := \{\, (\alpha \in \mathbb{N}_0)^{\mathbb{N}} | \text{only finitely many } \alpha_i \text{ are nonzero} \}. \tag{A.10}$$

The modulus and factorial of $\alpha \in (N_0)_c^{\mathbb{N}}$ are defined as $|\alpha| = \sum_{i \in \mathbb{N}} \alpha_i$ and as $\alpha! := \prod_{i \in \mathbb{N}} (\alpha_i!)$. The length of a multi-index is the largest i with $\alpha_i \neq 0$.

As Θ was assumed to be separable, it has a countable orthonormal basis of random variables $\theta = \{\theta_i\}_{i \in \mathbb{N}}$. The *multivariate Hermite polynomial* for a multi-index α is defined as

$$H_\alpha(\theta) := \prod_{i \in \mathbb{N}} h_{\alpha_i}(\theta_i). \tag{A.11}$$

This yields an explicit representation of the Wiener chaos: The set of all $H_\alpha(\theta)$ with $|\alpha| = k$ is an orthogonal basis of $\mathcal{H}_{=k}$. Hence, owing to the decomposition Eq. (A.9), any $\Sigma(\theta)$-measurable random variable on Ω with finite variance has an L^2 convergent approximation in the multivariate Hermite polynomials; this was shown by Cameron and Martin [23].

In numerical applications, stochastic quantities may be approximated in the polynomial chaos of order k over a finite-dimensional Gaussian Hilbert space

$H_{[0,0]}(\theta_1,\theta_2)$	$=$	1		$H_{[1,0]}(\theta_1,\theta_2)$	$=$	$h_1(\theta_1)$
$H_{[0,1]}(\theta_1,\theta_2)$	$=$	$h_1(\theta_2)$		$H_{[1,1]}(\theta_1,\theta_2)$	$=$	$h_1(\theta_1)h_1(\theta_2)$
$H_{[2,0]}(\theta_1,\theta_2)$	$=$	$h_2(\theta_1)$		$H_{[0,2]}(\theta_1,\theta_2)$	$=$	$h_2(\theta_2)$

Table A.2: Two-dimensional polynomial chaos of order 2.

$\Theta^{(m)}$ with orthonormal basis $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$. The computations may then be performed using the orthogonal basis of $\mathcal{H}_{\leq k}$ that consists of all $H_\alpha(\boldsymbol{\theta})$ with $|\alpha| \leq k$, e.g. see Table A.2. Consequently, the m-dimensional polynomial chaos of degree k is identified by the multi-index set

$$(\mathbb{N}_0)^m_{\leq k} := \{\alpha \in \mathbb{N}^m \,|\, |\alpha| \leq k\}. \tag{A.12}$$

For convenience, some properties of the polynomial chaos are collected below:

1. The vector space dimension of the polynomial chaos in m stochastic independent random variables is [e.g. 75, Corollary 3.24]

$$\mathcal{H}^m_{\leq k} = \binom{k+m}{m}. \tag{A.13}$$

   As Table A.3 shows, this number grows rapidly in the number of Gaussian random variables and in the polynomial degree.

2. The H_α are orthogonal and

$$\mathbf{E}\left(H_\alpha H_\beta\right) = \alpha!\,\delta_{\alpha\beta}, \quad \text{in particular} \quad \|H_\alpha\|^2_{L^2} = \alpha! \tag{A.14}$$

3. As the polynomial chaos is an orthogonal basis of $L^2 := L^2(\Omega, \Sigma(\Theta), P)$, any random variable of finite variance $f \in L^2$ has the L^2-convergent expansion

$$f = \sum_\alpha f^{(\alpha)} H_\alpha, \tag{A.15}$$

   where $f^{(\alpha)} = \frac{1}{\|H_\alpha\|^2_{L^2}} \mathbf{E}\,(f H_\alpha) = (\alpha!)^{-1} \mathbf{E}\,(f H_\alpha)$.

4. The projection $f^{(\alpha)}$ may be computed analytically for smooth random variables. Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ be an orthonormal basis of Θ and let a random variable $f \in L^2(\Omega, \Sigma(\Theta), P)$ be given as a function $f(\theta_1, \ldots, \theta_m)$. If all its partial derivatives belong to L^2, then

$$f^{(\alpha)} = (\alpha!)^{-1}\, \mathbf{E}\,(D^\alpha f), \tag{A.16}$$

   where D^α is the partial derivative with respect to the multi-index [e.g. 100, Theorem 3.1].

Stochastic dimensions m	Polynomial Degree k	Vector space dimensions of polynomial chaos
5	3	56
	5	252
10	3	286
	5	3 003
20	3	1 771
	5	$\approx$ 53 000
100	3	$\approx$ 177 000
	5	$\approx$ 96 000 000

Table A.3: Vector space dimensions of the polynomial chaos of degree k in m independent random variables.

A.3 Stochastic Distributions

Just as generalised functions (distributions) are continuous functionals on test spaces of smooth functions [43], generalised random variables (stochastic distributions) are functionals on test spaces of smooth random variables [70, 71]. The distribution spaces considered here are called the Hida and Kondratiev distribution spaces [15, 70, 71].

Analogous to the characterisation of tempered distributions by the Fourier coefficients of their Hermite expansion [e.g. 149, p.143], generalised random variables may be constructed as formal polynomial chaos expansions [70, 71]: Let the multivariate Hermite polynomials H_α on a separable Gaussian Hilbert space Θ be defined as in Eq. (A.11). Let V be a separable Hilbert space, let $\rho \in [-1,1]$ and $r \in \mathbb{R}$. For any (formal) expansion $f = \sum_\alpha f^{(\alpha)} H_\alpha$, with $f^{(\alpha)} \in V$ for all multi-indices α, define

$$\|f\|_{\rho,r}^2 := \sum_\alpha \|f^{(\alpha)}\|_V^2 \, (\alpha!)^{1+\rho} \, (2\mathbb{N})^{r\alpha}, \quad \text{where } (2\mathbb{N})^{r\alpha} := \prod_{j\in\mathbb{N}} (2j)^{r\alpha_j}, \quad \text{(A.17)}$$

and define $(S)^{\rho,r}$ as the vector space of all such f with $\|f\|_{\rho,r} < \infty$. Then $\|\cdot\|_{\rho,r}$ is a norm and the spaces $(S)^{\rho,r}$ are separable Hilbert spaces [70, 71] equipped with the scalar product

$$(f,g)_{\rho,r} := \sum_\alpha (f^{(\alpha)}, g^{(\alpha)})_V \, (\alpha!)^{1+\rho} \, (2\mathbb{N})^{r\alpha}. \quad \text{(A.18)}$$

The dual of $(S)^{\rho,r}$ may be identified with $(S)^{-\rho,-r}$ and the duality pairing on $(S)^{-\rho,-r} \times (S)^{\rho,r}$ is

$$\langle F, f \rangle := \sum_\alpha (F^{(\alpha)}, g^{(\alpha)})_V \, \alpha!. \quad \text{(A.19)}$$

It is obvious that $(S)^{0,0} = L^2(\Omega)$. For $\rho > 0, r > 0$, the random variables (RVs) $f \in (S)^{\rho,r}$ have coefficients $\| f^{(\alpha)} \|$ that decrease rapidly when the degree $|\alpha|$ grows or when the length of α (the maximum index i of non-zero elements α_i) grows. For $\rho, r > 0$ the spaces therefore contain RVs that have faster decreasing coefficients than required for finite variance. By analogy to the Fourier transform of deterministic functions, one may say that the larger ρ or the larger r, the more regular are the random variables in $(S)^{\rho,r}$. These spaces are test function spaces, similar to the space of rapidly decreasing functions. Their duals $(S)^{-\rho,-r}$ are the spaces of stochastic distributions or of generalised random variables. Members of $(S)^{-\rho,-r}$ are generalised RVs or stochastic distributions, i.e. linear functionals acting on the random test functions. For $\rho \in [0,1]$ the $(S)^{\rho} := \cap_{k\geq 0}(S)^{\rho,k}$ (with the projective limit topology) and their duals, the *Kondratiev distribution spaces* $(S)^{-\rho} := \cup_{k\geq 0}(S)^{-\rho,-k}$, may defined [71].

Now, approximations of RVs and generalised RVs in the polynomial chaos are considered. Let $f \in (S)^{\rho,q}$, and let $f^m_{\leq k}$ be its projection onto the m-dimensional polynomial chaos of degree k, i.e.

$$f^m_{\leq k} := \sum_{\substack{\alpha \in \mathbb{N}_0^m \\ |\alpha| \leq k}} f^{(\alpha)} H_\alpha. \tag{A.20}$$

An estimate for the error of this approximation was given by Benth and Gjerde [15]:

Theorem A.2: (Benth and Gjerde [15, Theorem 3.1])
Choose $p > 0$ and $q \in \mathbb{R}$ such that $r := p - q > r^*$, where $r^* \approx 1.53$ solves the equation $r^* = 2^{r^*}(r^* - 1)$. Let $\rho \in [-1,1]$. Then for any $f \in (S)^{-\rho,-q}$ and $g \in (S)^{\rho,p}$ the inequality

$$\left| \left\langle f - f^m_{\leq k}, g \right\rangle \right| \leq \| f \|_{-\rho,-q} \cdot \| g \|_{\rho,p} \cdot c(m,k,p-q) \tag{A.21}$$

holds, where

$$c(m,k,r) = \sqrt{c_1(r) m^{1-r} + c_2(r) \left(\frac{r}{2^r (r-1)} \right)^{k+1}} \tag{A.22}$$

and where $c_1(r) = (2^r(r-1) - r)^{-1}$ and $c_2(r) = 2^r(r-1) \cdot c_1(r)$.

Remark A.1: An inspection of the proof given in [15, Theorem 3.1] reveals that the estimate depends on $r := p - q$ being greater than r^*, and still holds for $p \in \mathbb{R}$. The proof concludes Eq. (A.22) from the inequality

$$\| f - f^m_{\leq k} \|_{-\rho,-p} \leq \| f \|_{-\rho,-q} \cdot c(m,k,p-q). \tag{A.23}$$

Example A.1: The following example is given in [15, Example 3.4]: If $p = q+2$, then $r = 2$ and $c(m,k,r)^2 = 1/2(m^{-1} + (1/2)^{k-1})$. Hence,

$$\|f - f^m_{\leq k}\|_{-\rho,-p} \leq \frac{1}{2}\|f\|_{-\rho,-p+2} \cdot (m^{-1} + 2^{-k+1}). \tag{A.24}$$

Appendix B

High-Dimensional Integration

This appendix extends the brief discussion on high-dimensional integration of Section 3.3.

The numerical techniques of this work require to evaluate high-dimensional integrals. For instance, random fields are discretised in Section 3.4.2 by orthogonal projection as $\kappa^{(\alpha)} = \mathbf{E}(\kappa H_\alpha)$ and the Galerkin projections for nonlinear problems in Section 3.3 involve integrals $\mathbf{E}\left(f(\kappa,u)H_\beta\right)$. In general, integrals

$$\begin{aligned} Q(\psi) &:= \mathbf{E}(\psi(\boldsymbol{\theta})) = \int_{\Omega^{(m)}} \psi(\boldsymbol{\omega})\, dP_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \\ &= \int_{\Omega_1} \cdots \int_{\Omega_m} \psi(\omega_1,\omega_2,\ldots,\omega_m) dP_{\theta_1}(\omega_1)\cdots dP_{\theta_m}(\omega_m) \end{aligned} \tag{B.1}$$

need to be evaluated. Several methods may be used for this and the following integration techniques are presented here: Monte Carlo methods, Quasi-Monte Carlo methods, full tensor product quadrature formulas, and Smolyak quadrature algorithms [163].

Each of these techniques computes an approximation $Q_Z(f)$ of Eq. (B.1) by evaluating the integrand in Z integration points $\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(Z)} \in \Omega^{(m)}$ and by linearly combining the results with weights $w_1,\ldots,w_Z \in \mathbb{R}$,

$$Q_Z(\psi) = \sum_{i=1}^{Z} w_i \psi(\boldsymbol{\omega}^{(i)}). \tag{B.2}$$

B.1 Monte Carlo Methods

For introductions to Monte Carlo methods see e.g. [40, 155] or Caflish's overview article [22] and the references therein.

Monte Carlo methods choose the integration points $\{\boldsymbol{\omega}^{(i)}\}$ as Z independent realisations of the random vector $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_m)^t$ and use $w_i = 1/Z$. The integral

is thus approximated as

$$Q_Z(\psi) = \frac{1}{Z}\sum_{i=1}^{Z}\psi(\boldsymbol{\omega}^{(i)}) \tag{B.3}$$

and the estimate $Q_Z(\psi)$ is a random variable converging almost surely to $Q(\psi)$ due to Kolmogorov's strong law of large numbers. For large Z, the error $\varepsilon_Z := |\mathbf{E}(\psi) - Q_Z(\psi)|$ (a random variable) is

$$\varepsilon_Z \approx \sigma Z^{-1/2}\mathcal{N}(0,1), \tag{B.4}$$

where $\mathcal{N}(0,1)$ is a standard Gaussian RV and where σ is the standard deviation of ψ. The error is probabilistic and hence predictions can only be made with some confidence level. An alternative convergence estimate based on the *Koksma-Hlawka theorem* is discussed in the next section.

Due to the slow convergence rate of order $O(\sigma Z^{-1/2})$ evaluations with high accuracy require a high computational effort and a reduction of σ is important. Monte Carlo methods may be sped up by various techniques for variance reduction. Some common techniques [22] are briefly mentioned here: *Antithetic Variables* add integration points at $-\boldsymbol{\omega}^{(i)}$, which reduces the variance as the linear term of the Taylor expansion of $\psi(\boldsymbol{\omega}^{(i)}) + \psi(-\boldsymbol{\omega}^{(i)})$ around zero has zero expectation. *Control Variates* compute $Q_Z(\psi - \phi)$, where ϕ is a function with known $\mathbf{E}(\phi)$. *Matching Moment Methods* modify the sequence $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(Z)}$ so that their statistical moments match the moments of the underlying distribution. *Stratification* computes the integral over $\Omega^{(m)}$ as the sum of integrals over disjoint sets partitioning $\Omega^{(m)}$ and may be enhanced by recursive application [142]. *Importance Sampling* writes the integral as $\mathbf{E}(\psi) = \int_{\Omega^{(m)}}[\psi(\boldsymbol{\omega})/p(\boldsymbol{\omega})]p(\boldsymbol{\omega})dP_{\boldsymbol{\omega}}(\boldsymbol{\omega})$, where p is a probability density similar to ψ. This is then interpreted as integration of $\psi(\boldsymbol{\omega})/p(\boldsymbol{\omega})$ with respect to the probability density $p(\boldsymbol{\omega})dP_{\boldsymbol{\omega}}(\boldsymbol{\omega})$, and the integration points are generated accordingly.

Monte Carlo simulations require reliable pseudo-random number generators, e.g. see [92] for an introduction. Inadequate random number generators produce biased results (e.g. due to artificial correlations in tuples of pseudo-random numbers). As a finite state machine, every pseudo-random number generator repeats itself after some number of iterations. Then the approximation error ceases to decrease. Hence, a random number generator must produce independent tuples and have a large cycle length. On a parallel computer, the sequences in the individual processes also need to be mutually independent.

According to Caflish [22], the pseudo-random number generators in [143, chapter 7] are reliable. In a review of parallel random generators [29] some packages are recommended for parallel random number generation, e.g. the SPRNG (Scalable Parallel Pseudo Random Number Generators) library [103] which permits long sequences and which is also recommended in [22].

B.2 Quasi-Monte Carlo Methods

Quasi-Monte Carlo methods evaluate the integrand in correlated points generated from so-called *low discrepancy series*. An important estimate for the upper bound in the error of an approximation of $Q(\psi) = \int_{[0,1]^m} \psi(\omega) d\omega$ computed by $Q_Z(\psi) = Z^{-1} \sum_{i=1}^{Z} \psi(\boldsymbol{\omega}^{(i)})$ from a series $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(Z)} \in [0,1]^m$ is the *Koksma-Hlawka theorem* [e.g. 22, Theorem 5.1]. It states that the integration error $\varepsilon = |I_m - Q_Z|$ is

$$\varepsilon \leq V(\psi) D_Z, \tag{B.5}$$

where $V(\psi)$ is the *total variation* of the integrand and D_Z is the *discrepancy* of the series $\{\boldsymbol{\omega}^{(i)}\}$, see e.g. [22] for the exact definitions. Intuitively speaking, the discrepancy is the maximal error in approximating volumes of rectangular sets inside $[0,1]^m$ by using samples from the series. Caflish [22] states that the total variation usually overestimates the error, while the discrepancy of the series is usually a good indicator for the actual error. A sequence $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(Z)}$ is called *quasi-random* if its discrepancy obeys

$$D_Z \leq c (\log Z)^n Z^{-1}, \tag{B.6}$$

where c, n are constants. They are independent of Z but usually depend on the dimension m. Often, $n = m$, and then the typical quasi-Monte Carlo error is obtained as $O(Z^{-1} \cdot (\log Z)^m)$. For high dimensions, the term $(\log Z)^m$ dominates, but for many integrands a convergence rate of $O(Z^{-1})$ is obtained [161].

Various quasi-random sequences have been developed, e.g. Halton's [143, Chapter 7.7] or Sobol sequences; see Niederreiter's monograph [121].

According to Caflish [22], quasi-Monte Carlo algorithms often converge faster compared to Monte Carlo in low dimensions, while their effectiveness reduces for large dimensions. For non-smooth integrands, they may become less effective, but their applicability may be increased by dimension reduction techniques. A comparison with quadrature is given by Schürer [161].

B.3 Quadrature by Full Tensor Products

Quadrature methods for high dimensions may be constructed as tensor products of one-dimensional quadrature formulas—e.g. of Gaussian, Clenshaw-Curtis, or Simpson quadrature formulas [e.g. 143, 160]. Assume that in each dimension Ω_i a quadrature formula $Q^{(i)}$ is given ($i = 1, \ldots, m$), each with the same number of nodes Z and each exactly integrating polynomials of degree k with respect to the measure $dP_{\theta_i}(\omega_i)$. The expectation $\mathbf{E}(\psi_i(\theta_i))$ of a function $\psi_i \colon \Omega_i \to \mathbb{R}$ may be

approximated as

$$Q^{(i)}(\psi_i) = \sum_{z=1}^{Z} w_i^{(z)} \psi_i(\omega_i^{(z)}), \tag{B.7}$$

where $w_i^{(z)}$ are the weights and $\omega_i^{(z)} \in \Omega_i$ are the nodes of the quadrature formula $Q^{(i)}, z = 1, \ldots, Z$.

A quadrature formula Q^m on $\Omega^{(m)}$ may be constructed as tensor product of the one-dimensional quadrature formulas, $Q^m := Q^{(1)} \otimes \cdots \otimes Q^{(m)}$. The integral $\psi : \Omega^{(m)} \to \mathbb{R}$ with respect to the measure $dP_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ may then be approximated as

$$Q^m(\psi) = \sum_{z_1=1}^{Z} \sum_{z_2=1}^{Z} \cdots \sum_{z_m=1}^{Z} w_1^{(z_1)} \cdots w_m^{(z_m)} \psi\big(\omega_1^{(z_1)}, \ldots, \omega_m^{(z_m)}\big). \tag{B.8}$$

This formula is exact for multivariate polynomials with (partial) degree up to k, where a multinomial $\omega_1^{\alpha_1} \cdot \cdots \cdot \omega_m^{\alpha_m}$ has (partial) degree k if $\alpha_i \leq k$ for all i.

The computation of Eq. (B.8) requires Z^m evaluations of the integrand. This is not feasible for high dimensions: for example, in $m = 30$ dimensions it requires more than a billion function evaluations if $z > 1$.

B.4 Smolyak Quadrature and Sparse Grids

Quadrature formulas based on Smolyak type combinations [163] of one-dimensional quadrature rules were applied successfully to high-dimensional integration, e.g. to 360-dimensional problems [137, 139]. Other names for such constructions are *sparse grid methods*, *Biermann interpolation*, *Boolean methods*, *discrete blending methods*, or *hyperbolic cross points*; see [45, 123, 125] and the references therein. The following exposition is based on [45, 125].

The Smolyak Construction: The Smolyak quadrature formula requires quadrature formulas $Q_1^{(i)}, Q_2^{(i)}, Q_3^{(i)}, \ldots$ in each dimension $\Omega_i, i = 1, \ldots, m$. Assume that every method $Q_l^{(i)}$ of level l exactly integrates polynomials of degree not exceeding k_l (independent of i) with respect to the measure P_{θ_i}, with $k_{l+1} \geq k_l$. Additionally, assume that each $Q_l^{(i)}$ has the same number of nodes Z_l, where all lowest order methods use only one node, $Z_1 = 1$. The set of nodes used by $Q_l^{(i)}$ will be called $\Xi_l^{(i)} = \{\omega_{l,1}^{(i)}, \ldots, \omega_{l,Z_l}^{(i)}\}$ and the weights will be denoted by $w_{l,1}^{(i)}, \ldots, w_{l,Z_l}^{(i)}$.

As discussed in the previous section, tensor products of quadrature formulas are not practical in high dimensions if they all use more than one node. But if the tensor product combines high order formulas in few dimensions with low order formulas in the other dimensions, then the resulting formula may be feasible in high dimensions.

For an $\boldsymbol{l} = (l, \ldots, l_m)^t \in \mathbb{N}^m$, a quadrature formula for functions $\psi\colon \Omega^{(m)} \to \mathbb{R}$ may be constructed as

$$Q_{\boldsymbol{l}} := Q_{l_1}^{(1)} \otimes \cdots \otimes Q_{l_m}^{(m)}$$

which is applied to ψ by

$$Q_{\boldsymbol{l}}(\psi) = \sum_{z_1=1}^{Z_{l_1}} \cdots \sum_{z_m=1}^{Z_{l_m}} w_{l_1,k_1}^{(1)} \cdots w_{l_m,z_m}^{(m)} \cdot \psi\big(\omega_{l_1,z_1}^{(1)}, \ldots, \omega_{l_m,z_m}^{(m)}\big).$$

As the lowest order quadrature formula uses $Z_1 = 1$ points, the evaluation of $Q_{\boldsymbol{l}}(\psi)$ is feasible even in high dimensions if $l_i \neq 1$ for only few i.

The Smolyak construction combines such formulas. Let $Q_0^{(i)} := 0$ for all i and

$$\Delta_l^{(i)} := Q_l^{(i)} - Q_{l-1}^{(i)}, \quad l \in \mathbb{N},\ i = 1, \ldots, m. \tag{B.9}$$

For $l \in \mathbb{N}_0$ the level l Smolyak quadrature formula in m dimensions is

$$S_l^m = \sum_{\boldsymbol{l} \in \mathbb{N}^m,\, |\boldsymbol{l}| \leq m+l-1} \Delta_{l_1}^{(1)} \otimes \cdots \otimes \Delta_{l_m}^{(m)}, \tag{B.10}$$

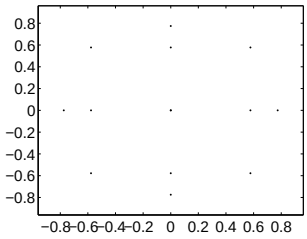
where $|\boldsymbol{l}| = l_1 + \cdots + l_m$. This may be rewritten [45, 123] as

$$S_l^m = \sum_{\boldsymbol{l} \in \mathbb{N}^m,\, l \leq |\boldsymbol{l}| \leq l+m-1} (-1)^{m+l-1-|\boldsymbol{l}|} \binom{m-1}{|\boldsymbol{l}|-l} \cdot Q_{\boldsymbol{l}} \tag{B.11}$$

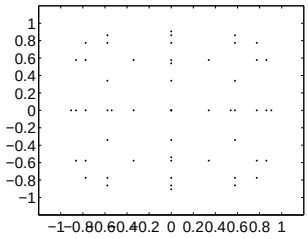
Sparse Grids: Every tensor product formula $Q_{\boldsymbol{l}}$ evaluates the integrand on a regular mesh of nodes $\boldsymbol{\Xi}_{\boldsymbol{l}} = \Xi_{l_1}^{(1)} \times \cdots \times \Xi_{l_m}^{(m)}$. The Smolyak formula evaluates the integrand on the union $\boldsymbol{\Xi}_l^m$ of these meshes

$$\boldsymbol{\Xi}_l^m := \bigcup_{|\boldsymbol{l}|_1 \leq m+l-1} \boldsymbol{\Xi}_{\boldsymbol{l}} = \bigcup_{|\boldsymbol{l}|_1 \leq m+l-1} \Xi_{l_1}^{(1)} \times \cdots \times \Xi_{l_m}^{(m)}. \tag{B.12}$$

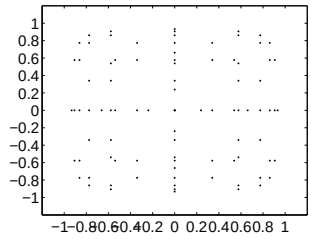
If the one-dimensional quadrature formulas are nested, i.e. if $\Xi_{l+1}^{(i)} \subseteq \Xi_l^{(i)}$, then $\boldsymbol{\Xi}_{\boldsymbol{l}} \subset \boldsymbol{\Xi}_{\boldsymbol{l}'}$ whenever $l_i \leq l_i', i = 1, \ldots, m$. This results in a smaller number of points compared to the non-nested formulas and hence in a reduced numerical effort. The resulting set of points $\boldsymbol{\Xi}_l^m$ may then be a *sparse grid*; see the examples discussed below and the plots for the Clenshaw-Curtis formulas in Fig. B.1. Explicit formulas for the number of nodes in such a sparse grid, and efficient algorithms for constructing it have been presented by Petras [139].



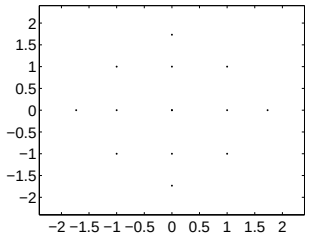
(a) Gauss-Legendre, S_3^2, 13 nodes.



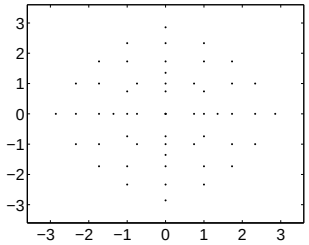
(b) Gauss-Legendre,S_5^2, 53 nodes.



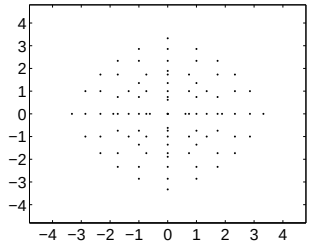
(c) Gauss-Legendre,S_6^2, 89 nodes.



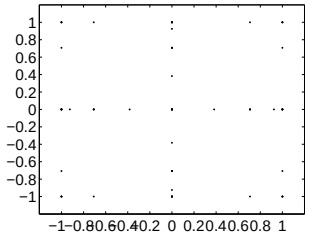
(d) Gauss-Hermite, S_3^2, 13 nodes.



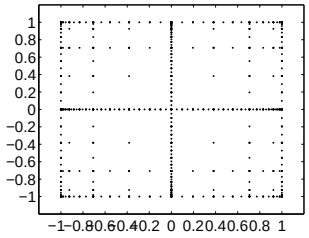
(e) Gauss-Hermite, S_5^2, 53 nodes.



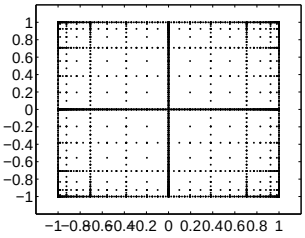
(f) Gauss-Hermite, S_6^2, 89 nodes.



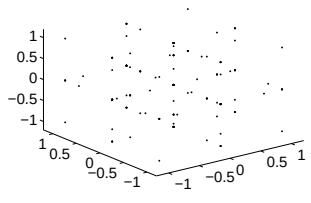
(g) Clenshaw-Curtis, S_4^2, 29 nodes.



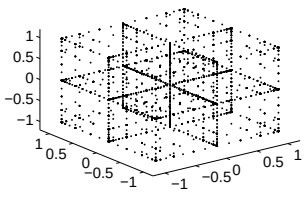
(h) Clenshaw-Curtis, S_7^2, 321 nodes.



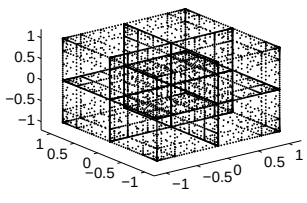
(i) Clenshaw-Curtis, S_9^2, 1517 nodes.



(j) Clenshaw-Curtis, S_4^3, 69 nodes.



(k) Clenshaw-Curtis, S_7^3, 1073 nodes.



(l) Clenshaw-Curtis, S_9^3, 5975 nodes.

Figure B.1: Grids for Smolyak quadrature.

If the univariate quadrature formulas $Q_l^{(i)}, i = 1, \ldots, m$, exactly integrate functions from spaces V_l, with $V_l \subset V_{l+1}$ for all l, then [123, Theorem 2] the Smolyak quadrature formulas S_l^m exactly integrate all functions from the space

$$\sum_{\boldsymbol{l} \in \mathbb{N}^m,\, |\boldsymbol{l}| = l} V_{l_1} \otimes \cdots \otimes V_{l_m}. \tag{B.13}$$

Example B.1: To integrate smooth functions ψ, a good choice may be to select $Q_l^{(i)}$ as the l-point Gauss formula corresponding to the measure P_{θ_i} [125].

For example, if $\boldsymbol{\theta}$ is uniformly distributed, then Gauss-Legendre formulas may be used (see the first row of Fig. B.1). If $\boldsymbol{\theta}$ is Gaussian, then Gauss-Hermite formulas may be used (see the second row of Fig. B.1).

As l-point Gauss formulas are exact for polynomials of degree at most $2l-1$, the resulting Smolyak formula S_l^m is exact for polynomials of total degree $2l-1$ according to Eq. (B.13), where a monomial $\omega_1^{\alpha_1} \cdots \omega_m^{\alpha_m}$ has *total degree* k if $\sum_i \alpha_i = k$.

Example B.2: Another common choice for the one-dimensional formulas are Clenshaw-Curtis formulas. The Z-point Clenshaw-Curtis (CC) formula is exact for polynomials of degree not exceeding Z.

CC formulas have lower integration order than Gauss formulas. Nonetheless, they are often used [e.g. 123, 139] as their nodes may be nested: If $Q_l^{(i)}$ is the CC formula in $Z_l := 2^{l-1}+1$ points and if $Q_1^{(i)}$ is the 1-point CC formula, then their nodes are nested and a sparse grid results for the Smolyak formulas S_l^m (see the last row in Fig. B.1). The resulting formulas have a total polynomial exactness of $2^{l-1}+1$. The costs in computing the weights may be reduced by storing them in a tree structure [137].

Schürer [161] compares adaptive and non-adaptive interpolatory quadrature rules to Monte Carlo and quasi-Monte Carlo methods for dimensions up to $m = 100$. He shows by various experiments how the dimensions and the integrand's regularity determine, which integration method performs best.

Smolyak constructions may not only be used for quadrature, but also for the interpolation of functions [60, 163], for the construction of finite element ansatz spaces [61, 159, 192], and for finite difference discretisations [154].

B.5 Conclusions

Of the discussed integration methods, Monte Carlo methods are suitable for low accuracy requirements and for integrands with small variance. They do not exploit

the integrand smoothness, and their advantage is their dimension independence. For efficiency, they need to be combined with variance reduction techniques.

Quasi-Monte Carlo methods may often have a better convergence rate than MC methods and they take low order smoothness into account. As Gerstner and Griebel [45] state, quasi-Monte Carlo methods may be advantageous compared to Smolyak integration if the integrands are not smooth.

Smolyak constructions are well suited for smooth integrands, and Gerstner and Griebel [45] state that they outperform both Monte Carlo and quasi-Monte Carlo for smooth functions, except for very high-dimensional problems.

Appendix C

A Simple Code Coupling Example for StoFEL

This appendix shows the complete code that is necessary to solve the nonlinear equation Eq. (5.1) in StoFEL. The equation to be solved is

$$a(\boldsymbol{\theta})u(\boldsymbol{\theta}) + b(\boldsymbol{\theta})u(\theta)^3 = f(\boldsymbol{\theta}). \tag{C.1}$$

Its interpretation and examples of solutions were discussed in Section 5.2.1.

C.1 The PDE-Class for the Nonlinear Spring

The code that implements this example in the StoFEL framework is simple. Nonetheless, it demonstrates all aspects of coupling a deterministic code with StoFEL, apart from implementing a *Mesh*-subclass. All code examples are MATLAB [106] programs.

The constructor creates an object of the class *cpde_0dnl1*, which implements Eq. (C.1) in StoFEL. It is shown in Program C.1 on the next page. The lines 2–3 decide that a nonlinear stationary SPDE is defined. There is only one spatial degree of freedom. Hence, the underlying mesh for the geometry is defined in line 5 as a vector of length 1 by creating an instance of the *cmesharr*-class, which represents random vectors. In line 7, the names of the parameters are defined. These will later allow to assign values to the parameters comfortably. The variable *pde.fields* in line 8 will be used to store realisations of the parameters. The variable *pde.bc* in line 9 is unused here; it is used when implementing a class for an SPDE to represent realisations of fields on the boundaries.

set_params(): This method is called to assign a realisation of the stochastic equation to the *PDE*-object. Its source code is simple for this example and is

```
 1 function pde = cpde_0dnl1(params)
 2   pde.is_nonlinear  = 1;          % classification
 3   pde.is_stationary = 1;          % classification
 4   pde.spatial_dim = 1;            % spatial dimensions
 5   pde.mesh = cmesharr( 1 );       % Mesh with one DOF
 6   pde.spatial_dof = 1;            % spatial DOF number
 7   pde.field_names={'c','a','f'};  % Field-names
 8   pde.fields   = {};
 9   pde.bc       = {};
10   pde = class(pde,'cpde_0dnl1',cpde);% return PDE-object
```

Program C.1: The constructor for the nonlinear spring example

```
1 function pde = set_params( pde, field_vals, bc_vals )
2   pde.fields   = field_vals;
3   pde.bc       = bc_vals;
```

Program C.2: The *set_params()*-method for the nonlinear spring example

shown in Program C.2. This method sets the values of the parameters and thus yields a realisation of the SPDE. The *bc_vals-member* variable in line 3 is required for compatibility with StoFEL and is not used in this simple example.

get_resid(): The *get_resid()*-method may be used only after the *set_params()*-method was called. It computes the residual and is implemented in a straightforward manner in Program C.3:

```
1 function [pde,r] = get_resid( pde, u );
2   r = (pde.fields.a.*u+pde.fields.b.*u.^3)-pde.fields.f;
```

Program C.3: The *get_resid()*-method for the nonlinear spring example

solve(): This method computes a realisation of the solution of the SPDE and may be called only after calling *set_params()*. It is shown in Program C.4. The solution is computed in lines 3–6 by a formula for cubic equations.

jacobian() The *jacobian()*-method computes the derivative with respect to *u* and is shown in Program C.5.

```
1 function [pde,u] = solve( pde )
2   a=pde.fields.a; b=pde.fields.b; f=pde.fields.f;
3   expr2=(9*sqrt(b).*f+sqrt(3).* ...
4         sqrt(4*a.^3+27.*b.*f.^2)).^(1/3);
5   u = -(2/3)^(1/3)*a./(sqrt(b).*expr2)+...
6        expr2./(2^(1/3)*3^(2/3).*sqrt(b));
```

Program C.4: The *solve()*-method for the nonlinear spring example

```
1 function [pde,jac] = jacobian( pde, u )
2  jac = pde.fields.a + 3.* pde.fields.b.*u.^2;
```

Program C.5: The *jacobian()*-method for the nonlinear spring example

C.2 Using the Nonlinear Spring Class

Above, the MATLAB-code implementing the *PDE*-class for the nonlinear spring example in StoFEL was presented. It is shown now, how Eq. (C.1) may be solved using this class.

The code-examples below are simple. Nonetheless, SPDEs are defined and solved in StoFELin the same manner. The main difference between solving this simple example and SPDEs in StoFEL is that SPDEs require more data in their definition and that more parameters affect their discretisation. Apart from this, the commands used to solve and to define SPDEs are the same as shown here.

Defining an SPDE-instance: The *cpde_0dnl1*-class presented above is used in Program C.6 on the next page to define an *SPDE*-object representing Eq. (C.1).

The PDE-object is instantiated in line 2. Following this, two random variables named *sfa* and *sfb* are defined in lines 5–11. Both are chosen as $\beta(1/2, 1/2)$-distributed. Their distributions are defined by the *cdistrib_gauss_beta*-object created on line 7 and by the type of the underlying random variables, which are defined as Gaussian in line 8. If one would set here *base_rvtype=1*, then the underlying random variables would be uniformly distributed. As *a* and *b* are random variables, the covariance is set to a constant (line 9). Lines 10 and 11 use the parameters set above to define the random variables. Random fields and random variables are always defined on *Mesh*-objects. As the mesh-object of the PDE was chosen as a vector of length 1, lines 10 and 11 define random variables.

In line 14, a *csfield_list*-object is created. Objects of this class hold all random fields of the SPDE and are used later to discretise these random fields. In lines 15–16, the random variables *sfa* and *sfa* are added to this list. Lines 17–19 define the parameters *a* and *b* of Eq. (C.1) as references to the random variables *sfa* and

```
 1 function spde = def_spde0d_nl1
 2   pde = cpde_0dnl1;             % The PDE object
 3
 4   % define random variables a and b
 5   params.add      = .1;
 6   params.scale    = 0.5;
 7   distrib         = cdistrib_gauss_beta( params );
 8   base_rvtype     = 0;          % Gaussian RV
 9   cov = '1';
10   sfa = csfield('a',pde.mesh,cov,base_rvtype,distrib);
11   sfb = csfield('b',pde.mesh,cov,base_rvtype,distrib);
12
13   % Add random variables to the list of random fields
14   sf_list = csfield_list;
15   [sf_list,  sfa_ref] = add( sf_list, sfa );
16   [sf_list,  sfb_ref] = add( sf_list, sfb );
17   fields.a = sfa_ref;
18   fields.b = sfb_ref;
19   fields.f = cfield( 'f', pde.mesh, 1); %deterministic
20
21   % Define SPDE
22   spde.pde     = pde;
23   spde.fields  = fields;
24   spde.bc      = {};     % no boundary conditions
25   spde.sf_list = sf_list;
```

Program C.6: Defi nition of an SPDE object for the nonlinear spring example

sfa. Line 19 sets the parameter 'f' of Eq. (C.1) to be a deterministic constant with value 1. Finally, lines 22–25 store all these data in the SPDE-object.

Using the SPDE-object: It is shown below how this SPDE-object may be used:

```
1 spde = def_spde0d_nl1;
2 [spde.omega,spde.sf_list]=discretise(spde.sf_list);
```

On line 1, the SPDE-object is instantiated by calling *def_spde0d_nl1* defined in Program C.6. In line 2, the stochastic space is discretised using the default settings and the object *spde.omega* is obtained, which describes the probability space. For a real SPDE, one would beforehand configure in an options-database how many terms of the Karhunen–Loève expansions to keep for each random field, but this is not necessary for this simple example.

Solving the SPDE in a Monte Carlo Fashion: Once the random fields are

discretised, the mean and the variance of the solution may be computed in a Monte Carlo fashion. The following code fragment is used for this:

```
 3 quad_opts.type = 'SM';
 4 quad_opts.quad_rule = 'GS';
 5 quad_opts.quad_stages = 7;
 6 quadu=omega_quad( spde.omega, quad_opts, ...
 7                   'spde_eval_solution', {spde} );
 8 u_mean = quadu.mean
 9 u_var  = quadu.var
```

Lines 3–5 choose the integration method. The 'SM' in line 3 stands for Smolyak quadrature. One might instead also specify 'FT' for full tensor product quadrature, or 'MC' for Monte Carlo integration.

Line 4 selects the one-dimensional quadrature rule used to compose the high-dimensional quadrature rule. Here, 'GS' indicates that a Gauss quadrature rule is used. By specifying 'CC' instead, one might also use nested Clenshaw-Curtis quadrature rules. Line 5 chooses the order of integration and lines 6–7 perform the high-dimensional integration: the command *omega_quad* integrates the results of the function given in its third argument. The function that is integrated over the probability space is here *spde_eval_solution*. It computes realisations of solutions for the SPDE-object. Lines 8–9 retrieve the mean and the variance of the solution.

Polynomial Chaos solution: A Galerkin solution in generalised polynomial chaos requires more configuration steps:

```
10 spde.ansatz = ansatz_pcss_create( spde.omega, 4);
11 bfgs.max_iter  = 20;
12 bfgs.tolerance = sqrt(eps);
13 pde = set_params( pde, qsf.mean, {}, 1 );
14 qJ = omega_quad( spde.omega, quad_opts, ...
15                  'spde_eval_bdiag', {spde,u});
16 bfgs.H0  = diag(qJ.mean)
17 [u,bfgs] = bfgs_solve( bfgs, ...
18                        spde.ansatz.u, ...
19                        'spde_pc_resid',     ...
20                        {spde,quad_opts});
21 u_mean = get_mean(u);
22 u_var  = get_var(u);
```

In line 10, a generalised polynomial chaos ansatz of degree 4 is created. Here, the orthogonal ansatz functions are automatically chosen according to the types of the independent random variables θ_1, θ_2 used in the SPDE-definition.

Lines 11–16 configure the BFGS-solver. Lines 14-15 compute the block-diagonal preconditioner by integrating the Jacobian over the probability space. Storing the preconditioner as a matrix is possible for this simple example. For true SPDEs with many spatial degrees of freedom it is usually not possible to store the preconditioner as a matrix. Instead, the BFGS-solver may be configured to call the deterministic solver as a preconditioner when solving SPDEs.

The BFGS-solver is executed in lines 17–19. It computes the response surface u. The mean and the variance of u are computed in lines 21–22. Other statistics of the solution may now be computed by sampling from the response surface.

See Section 5.2.1 for solutions obtained by executing the code shown in this appendix.

Appendix D

Terms and Symbols

A consistent notation is used in the text. A glossary may be found in Appendix D.1, notational conventions are discussed in Appendix D.2 and a commented list of symbols is shown in Appendix D.3. See also the index on page 170.

D.1 Glossary

The following abbreviations are used:

CC	Clenshaw-Curtis (quadrature rule)
CDF	Cumulative distribution function (of a random variable)
Deterministic Code	See deterministic solver
Deterministic Solver	It is assumed that the spatial discretisation is performed by some existing simulation software, which is called the *deterministic solver* or the *deterministic code*
DOF	Degree of freedom
FE	Finite element
FEM	Finite element method
FORM	First order reliability method
KL	Karhunen–Loève (e.g. KL-expansion, KL-series)
PDE	Partial differential equation
PDF	Probability density function
RF	Random field
RV	Random variable
SFEM	Stochastic finite element method
SORM	Second order reliability method
SPDE	Stochastic partial differential equation

D.2 Notation and Conventions

The following general conventions are used:

$\boldsymbol{u}$	Vectors are written in small bold italic letters.
$\mathbf{u}$	Block vectors are written in small bold upright letters.
$\boldsymbol{K}$	Matrices are written in capital bold italic letters.
$\mathbf{K}$	Block matrices are written in capital bold upright letters.
γ, κ, ξ	Random variables and random fields are written in Greek letters.
$\boldsymbol{\gamma}, \boldsymbol{\kappa}, \boldsymbol{\xi}$	Random vectors are written in bold Greek letters.
α, β, ι	These Greek letters denote multi-indices.
$\kappa^{(\alpha)}$	A superscript multi-index denotes the coefficient of a random variable in its polynomial chaos expansion.
$\kappa^m_{\leq k}$	This is the projection of a random variable onto the m-dimensional polynomial chaos of degree k.

The following conventions are used for random variables and random fields:

$u(\omega)$	This is the random variable u expressed as a map on the space of elementary events, $u\colon \Omega \to \mathbb{R}$.
$u(\theta)$	This is the random variable u expressed as a function of the mutually independent basic random variables $\theta = (\theta_1, \theta_2, \ldots)$, with $\theta_i : \Omega \to \mathbb{R}$. This is a short form for $u(\theta(\omega))$.
$u(\boldsymbol{\theta})$	Approximations of the random variable u in a finite subset $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ of the mutually independent basic random variables $\theta = (rv_1, \theta_2, \ldots)$ are written as $u(\boldsymbol{\theta})$.
$u(\boldsymbol{\omega})$	A realisation of the random variable u, where $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m) \in \Omega^{(m)}$ denotes a realisation of the random vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$.

D.3 Symbols

The following symbols are used in the text:

$\|\cdot\|_p$	Standard L^p-norm (Appendix A.2.1)
$\|\cdot\|_\infty$	Standard L^∞-norm, essential supremum (Appendix A.2.1)
$\|\cdot\|_{\rho,r}$	Hida distribution and test function norms (Appendix A.3)
$\|\cdot\|_\rho$	Kondratiev distribution and test function norms (Appendix A.3)
$\langle\cdot,\cdot\rangle$	Duality pairing (Appendix A.2.2)
$(\cdot,\cdot)$	Scalar product (Appendix A.2.2)
$\|\alpha\|$	Modulus of a multi-index, $\|\alpha\| = \sum_{i\in\mathbb{N}} \alpha_i$ (Appendix A.2.3)
$\alpha!$	Factorial of a multi-index, $\alpha! := \prod_{i\in\mathbb{N}} \alpha_i!$ (Appendix A.2.3)
$\odot$	The Wick product (Section 2.3.2)
α,β,ι	Multi-indices (Appendix A.2.3)
$\mathcal{B}$	σ-algebra of events of the probability space (Appendix A.1)
χ_B	Characteristic function of a set B
$C_c^\infty(R)$	Smooth functions with compact support in $R \subset \mathbb{R}^d$
cov_κ	Covariance function of RF (Section 2.2)
$\boldsymbol{C}_\kappa$	The covariance matrix of a random vector $\boldsymbol{\kappa}(\omega)$
d	Dimension of the spatial domain $R \subset \mathbb{R}^d$ (Section 2.2.1)
Δ	The Laplace operator
$\Delta^{(\iota)}$	A matrix $\Delta^{(\iota)}_{\alpha,\beta} := \mathbf{E}\left(H_\alpha(\theta)H_\beta(\theta)H_\iota(\theta)\right), \alpha,\beta \in \mathcal{I}$ (Section 4.1)
$\mathrm{erf}(x)$	Distribution function of a standard Gaussian RV (Appendix A.1)
$\mathbf{E}(\cdot)$	Expectation operator, $\mathbf{E}(g(\kappa)) = \int_\Omega g(\kappa(\omega))\, dP(\omega)$ (Appendix A.1)
$f_\kappa(x)$	Probability density function of real-valued RV (Appendix A.1)
$F_\kappa(k)$	Probability distribution function of real-valued RV (Appendix A.1)
γ	Gaussian RV or Gaussian RF $\gamma(x,\omega), x \in R, \omega \in \Omega$ (Section 2.2.3)
$h_i(x)$	Hermite-polynomial of degree i (Appendix A.2.3)
$H_\alpha(\omega)$	Multivariate Hermite-polynomial for multi-index α (Appendix A.2.3)
$\mathring{H}^1(R)$	Sobolev Hilbert space of weakly differentiable functions
$\mathcal{H}_{=p}$	Homogeneous chaos of degree p (Appendix A.2.3)
$\mathcal{H}^m_{=p}$	m-dimensional homogeneous chaos of degree p (Appendix A.2.3)
$\mathcal{H}_{\leq p}$	Polynomial chaos of degree p (Appendix A.2.3)
$\mathcal{H}^m_{\leq p}$	m-dimensional polynomial chaos of degree p (Appendix A.2.3)
$\mathcal{I}$	Identifies the stochastic ansatz, $\mathcal{I} \subset (\mathbb{N}_0)^{\mathbb{N}}_c$ (Section 3.4)

$\mathcal{J}(\mathcal{I})$	Identifies expansion of operator, $\mathcal{J}(\mathcal{I}) \subset (\mathbb{N}_0)_c^{\mathbb{N}}$ (Section 4.1)
κ	Random field $\kappa(x,\omega), x \in R, \omega \in \Omega$ (Section 2.2.1)
$\kappa^{(\alpha)}$	Projection of RF onto H_α (Section 3.4)
$\kappa_1, \kappa_2, \ldots$	Karhunen–Loève eigenfunctions, $\kappa_i \in L^2(R)$ (Section 3.1)
$\boldsymbol{K}(\theta)$	Stochastic system matrix (Section 3.2)
$\boldsymbol{K}^{(\iota)}$	Polynomial chaos projection of system matrix (Section 4.1.1)
$\boldsymbol{K}_i$	System matrix for i-th KL-eigenmode as material (Section 4.1.2)
$\mathbf{K}^l$	The l-term truncated KL-expansion of system matrix (Section 4.1.2)
$\mathbf{K}$	Block matrix $\mathbf{K} = \left(\boldsymbol{K}_{\alpha,\beta}\right)_{\alpha,\beta\in\mathcal{I}}$ of Galerkin method (Section 3.4.3)
λ_i	KL-eigenvalues, $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ (Section 4.1.2)
μ_κ	Mean of RV or RF κ (Appendix A.1)
$\boldsymbol{\mu}_{\boldsymbol{\kappa}}$	Mean vector of the random vector $\boldsymbol{\kappa}$
m	Number of independent random variables
n	Size of spatial ansatz (Section 3.2)
$\boldsymbol{N}(x)$	Spatial ansatz functions, $\boldsymbol{N}(x) = (N_1(x), \ldots, N_n(x))$ (Section 3.2)
$\mathcal{N}(\mu, \sigma^2)$	Gaussian RV with mean μ and variance σ^2) (Appendix A.1)
$(\mathbb{N}_0)_c^{\mathbb{N}}$	Space of multi-indices $(\alpha \in \mathbb{N}_0)^{\mathbb{N}}$ (Appendix A.2.3)
$(\mathbb{N}_0)^m_{\leq p}$	Set of multi-indices $\alpha \in \mathbb{N}^m$ with $\lvert\alpha\rvert \leq p$ (Appendix A.2.3)
$(\Omega, \mathcal{B}, P)$	A probability space (Appendix A.1)
ω	Elementary event $\omega \in \Omega$
Ω	Set of elementary events (Appendix A.1)
$\boldsymbol{\omega}$	An element $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m) \in \Omega^{(m)}$ (Section 3.1.5)
ω_i	An element $\omega_i \in \Omega_i$, where Ω_i is the range of θ_i (Section 3.1.5)
$\Omega^{(m)}$	Image of the RVs $\boldsymbol{\theta}$, $\Omega^{(m)} = \Omega_1 \times \cdots \times \Omega_m$, (Section 3.1.5)
Ω_i	Range of θ_i, $\Omega_i := \operatorname{range}(\theta_i) = \theta_i(\Omega)$ (Section 3.1.5)
$\boldsymbol{\omega}^{(i)}$	Integration point for the integration in $\Omega^{(m)}$ (Appendix B)
P	Probability measure (Appendix A.1)
P_m	Probability distribution of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ (Section 3.1.5)
P_κ	Probability distribution of RV κ (Appendix A.1)
$\mathcal{P}_n(G)$	Vector space of polynomials on Hilbert space G (Appendix A.2.3)
$Q_Z(\psi)$	Integration formula in Z integration points (Section 3.3)
$\rho_\kappa(x,y)$	Correlation function of RF, $\rho_\kappa(x,y) := \frac{\operatorname{cov}_\kappa(x,y)}{\sqrt{\operatorname{var}_\kappa(x)\operatorname{var}_\kappa(y)}}$
$\boldsymbol{r}(\boldsymbol{u}(\theta)$	Residual of semi-discretised SPDE (Section 3.2)
R	Spatial domain, $R \subset \mathbb{R}^d$ (Section 2.2.1)

σ_κ	Standard deviation of RV κ (Appendix A.1)
$\Sigma(\theta)$	σ-algebra generated by the $\theta = (\theta_1, \theta_2, \ldots)$
(S)	Space of admissible stochastic functions (Chapter 2)
$(S)^{\rho,r}$	Hida distribution and test function spaces (Appendix A.3)
$(S)^{\rho}$	Kondratiev distribution and test function spaces (Appendix A.3)
$(S)^{\mathcal{I}}$	Space of stochastic ansatz functions (Section 3.4)
$S(\mathbb{R}^d)$	Space of rapidly decreasing functions (Appendix A.2.2)
$S(\mathbb{R}^d)'$	Space of tempered distributions (Appendix A.2.2)
S_l^m	Smolyak formula of level l in m dimensions (Eq. (B.4))
θ	Sequence of independent RVs, $\theta = (\theta_1, \theta_2, \ldots)$ (Section 3.1.5)
$\boldsymbol{\theta}$	Vector of independent RVs, $\theta = (\theta_1, \ldots, \theta_m)$ (Section 3.1.5)
u	SPDE solution (Chapter 2)
u^h	Solution of semi-discretised SPDE, $u^h \in V^h \otimes (S)$ (Section 3.2)
$u^{h,\mathcal{I}}$	Series ansatz for SPDE solution, $u^h \in V^h \otimes (S)^{\mathcal{I}}$ (Section 3.4)
$\boldsymbol{u}(\theta)$	Coefficient vector of semi-discretisation (Section 3.2)
$\mathbf{u}$	Coefficient block vector of Galerkin discretisation (Section 3.4)
var_κ	Variance of RV κ
V	Admissible space of spatial functions (Chapter 2)
V^h	Space of finite element shape functions $V^h \subset V$ (Section 3.2)
w_i	Weights $w_1, \ldots, w_Z$ in an integration rule (Appendix B)
ξ_i	Uncorrelated RVs in Karhunen–Loève expansion (Section 3.1)
$\xi_i^{(\iota)}$	Projection of $\xi_i(\theta)$ onto H_ι, $\xi_i(\theta) = \sum_\iota xi_i^{(\iota)} H_\iota(\theta)$ (Section 3.4)
x	Point in spatial domain, $x \in R \subset \mathbb{R}^d$
y	Point in spatial domain, $y \in R \subset \mathbb{R}^d$
Z	Number of integration points (Appendix B)

Index

List of Figures

List of Tables

List of Algorithms

List of Programs

References

[1] ADLER, R. J.: *The Geometry of Random Fields*. John Wiley & Sons, Chichester, 1981.

[2] Advanced Visual Systems LTD., Surrey, England: *AVS User's Guide*. http://www.avs.com/, 1989–2003.

[3] ALEFELD, G. and J. MANNHEIMER: *Einführung in die Intervallrechnung*. Academic Press, New York, NY, 1974.

[4] ANSYS, Inc., Canonsburg, PA: *ANSYS Finite Element Software*. http://www.ansys.com.

[5] ANSYS, Inc., Canonsburg, PA: *Format of binary ANSYS data files*. http://www.ansys.com, 2001.

[6] ANSYS, Inc., Canonsburg, PA: *Guide to ANSYS user programmable features*. http://www.ansys.com, 2001.

[7] ATKINSON, K. E.: *The Numerical Solution of Integral Equations of the Second Kind*. Cambridge University Press, Cambridge, 1997.

[8] BABUŠKA, I. and P. CHATZIPANTELIDIS: *On solving linear elliptic stochastic partial differential equations*. Computer Methods in Applied Mechanics and Engineering, 191:4093–4122, 2002.

[9] BABUŠKA, I. and J. CHLEBOUN: *Effects of uncertainties in the domain on the solution of Neumann boundary value problems in two spatial dimensions*. Mathematics of Computation, 71(240):1339–1370, 2001.

[10] BABUŠKA, I. and K.-M. LIU: *On solving stochastic initial-value differential equations*. Mathematical Models & Methods in Applied Sciences, 13(5):715–745, 2003.

[11] BABUŠKA, I.; K.-M. LIU; and R. TEMPONE: *Solving stochastic partial differential equations based on the experimental data*. TICAM Report 02-18, Texas Institute for Computational and Applied Mathematics, University of Texas, Austin, TX, http://www.ticam.utexas.edu/reports/2002/0218.pdf, 2002.

[12] BABUŠKA, I.; R. TEMPONE; and G. E. ZOURARIS: *Galerkin finite element approximations of stochastic elliptic partial differential equations*. TICAM Report 02-38, Texas Institute for Computational and Applied Mathematics, University of Texas, Austin, TX, http://www.ticam.utexas.edu/reports/0238.pdf, 2002.

[13] BALAY, S.; K. BUSCHELMAN; W. D. GROPP; D. KAUSHIK; M. KNEPLEY; L. C. MCINNES; B. F. SMITH; and H. ZHANG: *PETSc user man-*

ual. Technical Report ANL-95/11, Argonne National Laboratory, `http://www.mcs.anl.gov/petsc`, 2002.

[14] BAUER, H.: *Wahrscheinlichkeitstheorie*. De Gruyter, Berlin, 1991.

[15] BENTH, F. E. and J. GJERDE: *Convergence rates for finite element approximations for stochastic partial differential equations*. Stochastics and Stochastic Reports, 63:313–326, 1998.

[16] BESOLD, P.: *Solutions to Stochastic Partial Differential Equations as Elements of Tensor Product Spaces*. Doctoral thesis, Georg-August-Universität, Göttingen, 2000.

[17] BOURGUND, U. and C. G. BUCHER: *Importance sampling procedure using design points—ISPUD, a user's manual*. Report 8-86, Institute of Engineering Mechanics, University of Innsbruck, cited in [19], 1986.

[18] BUCHER, C. and U. BOURGUND: *A fast and efficient response surface approach for structural reliability problems*. Structural Safety, 7:57–66, 1990.

[19] BUCHER, C.; D. HINTZE; and D. ROOS: *Stochastics and finite elements—challenges and chances*. In *Proceedings of the 17th CAD-FEM User's meeting*, Sonthofen, Germany, 1999.

[20] BUCHER, C.; D. HINTZE; and D. ROOS: *Advanced analysis of structural reliability using commercial FE-codes*. In *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS*, Barcelona, Spain, 2000.

[21] BUCHER, C. G.; Y. SCHORLING; and W. WALL: *SLang—the structural language, a tool for computational stochastic structural analysys*. In *Proceedings of the 10th ASCE Engineering Mechanics Conference*, 1132–1126, Boulder, CO, cited in [19], 1995.

[22] CAFLISCH, R. E.: *Monte Carlo and quasi-Monte Carlo methods*. Acta Numerica, 7:1–49, 1998.

[23] CAMERON, R. and W. MARTIN: *The orthogonal development of nonlinear functions in series of Fourier-Hermite functionals*. Annals of Mathematics, 48:385, 1947.

[24] CARIO, M. and B. NELSON: *Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix*. Technical report, Northwestern University, Evanston, IL, `http://citeseer.nj.nec.com/cario97modeling.html`, 1997.

[25] CHOQUET-BRUHAT, Y. and C. D. WITT-MORETTE: *Analysis, Manifolds and Physics*. North Holland, Amsterdam, 1982.

[26] CHRISTAKOS, G.: *Random field models in earth sciences*. Academic Press, New York, NY, 1992.

[27] CIARLET, P. G.: *The finite element method for elliptic problems*. North Holland, Amsterdam, 1978.

[28] CIORANESCU, D. and P. DONATO: *An Introduction to Homogenization*. Oxford University Press, Oxford, 1999.

[29] CODDINGTON, P. D.: *Random number generators for parallel computers*. The NHSE Review Vol. 2(2), National HPCC Software Exchange, Center for Research on Parallel Computation, Rice University, Houston, TX, `http://nhse.cs.rice.edu/NHSEreview/RNG/`, 1996.

[30] COHEN, A. and J.-P. D'ALES: *Nonlinear approximation of random functions*. SIAM Journal of Applied Mathematics, 57(2):518–540, 1997.

[31] COMSOL AB, Stockholm, Sweden: *FEMLAB—Multiphysics in Matlab*. `http://www.femlab.com/`.

[32] Cray Inc., Seattle, WA: *Cray Standard C and Cray C++ Reference Manual, No. 004–2179–005*. `http://www.cray.com/craydoc/20/`, 2000.

[33] DAGAN, G. and S. P. NEUMAN (editors): *Subsurface Flow and Transport: A Stochastic Approach*. Cambridge University Press, Cambridge, 1997.

[34] DEB, M. K.; I. BABUŠKA; and J. T. ODEN: *Solution of stochastic partial differential equations using Galerkin finite element techniques*. Computer Methods in Applied Mechanics and Engineering, 190:6359–6372, 2001.

[35] DENNIS, JR., J. E. and R. B. SCHNABEL: *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, Philadelphia, PA, 1996.

[36] DOOB, J. L.: *Stochastic Processes*. John Wiley & Sons, Chichester, 1953.

[37] ELISHAKOFF, I. (editor): *Whys and Hows in Uncertainty Modelling—Probability, Fuzziness and Anti-Optimization*. Springer, Berlin, 1999.

[38] ELMAN, H.; O. G. ERNST; D. P. O'LEARY; and M. STEWART: *Efficient iterative algorithms for the stochastic finite element method with applications to accoustic scattering*. Technical report, Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, College Park, MD, 2002.

[39] EVANS, L. C.: *Partial Differential Equations*. American Mathematical Society, Providence, RI, 1998.

[40] FISHMAN, G. S.: *Monte Carlo: Concepts, Algorithms, and Applications*. Springer, Berlin, 1999.

[41] Free Software Foundation, Boston, MA: *GNU Compiler Collection*. `http://gcc.gnu.org/`.

[42] FRINGS, W.: *The Cray Systems at the Research Centre Jülich, User's Manual BHB-0138*. Forschungszentrum Jülich GmbH, Jülich, `http://www.fz-juelich.de/nic/Supercomputer/computer-e.html`, 2002.

[43] GEL'FAND, I. M. and G. E. SHILOV: *Generalized Functions—Volume 1: Properties and operations*. Academic Press, New York, NY, 1964.

[44] GEL'FAND, I. M. and N. Y. VILENKIN: *Generalized Functions—Volume 4: Applications of harmonic analysis*. Academic Press, New York, NY, 1964.

[45] GERSTNER, T. and M. GRIEBEL: *Numerical integration using sparse grids*. Numerical Algorithms, 18:209–232, 1998.

[46] GHANEM, R.: *Hybrid stochastic finite elements: coupling of spectral expansions with Monte Carlo simulations*. Journal of Applied Mechanics, 65:1004–1009, 1998.

[47] GHANEM, R.: *Probabilistic characterization of transport in heterogeneous media*. Computer Methods in Applied Mechanics and Engineering, 158:199–220, 1998.

[48] GHANEM, R.: *Subsurface hydrology—scales of fluctuations and the propagation of uncertainty in random porous media*. Water resources research, 34(9):2123–2136, 1998.

[49] GHANEM, R.: *Ingredients for a general purpose stochastic finite elements implementation*. Computer Methods in Applied Mechanics and Engineering, 168(1–4):19–34, 1999.

[50] GHANEM, R.: *The nonlinear Gaussian spectrum of log-normal stochastic processes and variables*. Journal of Applied Mechanics, 66(4):964–973, 1999.

[51] GHANEM, R.: *Stochastic finite elements for heterogeneous media with multiple random non-Gaussian properties*. Journal of Engineering Mechanics, 125(1):24–40, 1999.

[52] GHANEM, R. and R. KRUGER: *Numerical solutions of spectral stochastic finite element systems*. Computer Methods in Applied Mechanics and Engineering, 129(3):289–303, 1996.

[53] GHANEM, R. and M. PELLISSETTI: *Adaptive data refinement in the spectral stochastic finite element method*. Communications in numerical methods in engineering, 18:141–151, 2002.

[54] GHANEM, R. and J. RED-HORSE: *Propagation of uncertainty in complex physical systems using a stochastic finite element approach*. Physica D, 133:137–144, 1999.

[55] GHANEM, R. and A. SARKAR: *Structural acoustics analysis of stochastic systems*. In *Proceedings of the European Congress on Computational Methods in Applied Science and Engineering, ECCOMAS 2000*, Barcelona, Spain, 2000.

[56] GHANEM, R. and P. SPANOS: *Spectral stochastic finite-element formulation for reliability analysis*. Journal of Engineering Mechanics, 117(10):2351–2372, 1991.

[57] GHANEM, R. and P. SPANOS: *Stochastic finite elements—A spectral approach*. Springer, Berlin, 1991.

[58] GHIOCEL, D. and R. GHANEM: *Stochastic finite-element analysis of seismic soil-structure interaction*. Journal of Engineering Mechanics, 128:66–77, 2002.

[59] GOLLWITZER, S.; A. ZVEREV; R. CUNTZE; and M. GRIMMELT: *Structural reliability applications in aerospace engineering*. In *Proceedings ICOSSAR '93 Innsbruck*, 1265–1272, Balkema, Rotterdam, cited in [20]. Implemented in the StruRel software, see `http://www.strurel.de`, 1994.

[60] GRIEBEL, M. and S. KNAPEK: *Optimized tensor-product approximation spaces*. Constructive Approximation, 16(4):525–540, 2000.

[61] GRIEBEL, M.; P. OSWALD; and T. SCHIEKOFER: *Sparse grids for boundary integral equations*. Numerische Mathematik, 83(2):279–312, 1999.

[62] GRIGORIU, M.: *Applied non-Gaussian processes: examples, theory, simulation, linear random vibration, and Matlab solutions*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[63] GRIGORIU, M.: *Stochastic Calculus—Applications in Science and Engineering*. Birkhäuser, Basel, 2002.

[64] GROPP, W.; E. LUSK; and A. SKJELLUM: *Using MPI: Portable Parallel Programming with the Message Passing Interface*. The MIT Press, Cambridge, MA, 1996.

[65] GROSSMANN, C. and H.-G. ROOS: *Numerik partieller Differentialgleichungen*. B.G. Teubner, Stuttgart, 1994.

[66] HACKBUSCH, W.: *Integralgleichungen, Theorie und Numerik*. B.G. Teubner, Stuttgart, 1995.

[67] HALDAR, A. and S. MAHADEVAN: *Reliability assessment using stochastic finite element analysis*. John Wiley & Sons, Chichester, 2000.

[68] HELMIG, R.: *Multiphase Flow and Transport Processes in the Subsurface*. Springer, Berlin, 1997.

[69] HENDERSON, S.; B. CHIERA; and R. COOKE: *Generating "dependent" quasi-random numbers*. In *Proceedings of the 2000 Winter Simulation Conference*, 527–536, Orlando, Fl, `http://citeseer.nj.nec.com/henderson00generating.html`, 2000.

[70] HIDA, T.; H.-H. KUO; J. POTTHOFF; and L. STREIT: *White Noise—An infinite dimensional calculus*. Kluwer, Dordrecht, 1993.

[71] HOLDEN, H.; B. ØKSENDAL; J. UBØE; and T.-S. ZHANG: *Stochastic Partial Differential Equations*. Birkhäuser, Basel, 1996.

[72] HUANG, S.; S. QUEK; and K. PHOON: *Convergence study of the truncated Karhunen-Loève expansion for simulation of stochastic processes*. International Journal for Numerical Methods in Engineering, 52:1029–1043, 2001.

[73] HUYSE, L. and M. A. MAES: *Stochastic finite element analysis using micro-mechanically based stochastic homogenization.*. In N. JONES and R. GHANEM (editors), *Proceedings of the 13th ASCE Engineering Mechanics Division Conference*, The Johns Hopkins University, Baltimore, MD, 1999.

[74] HUYSE, L. and M. A. MAES: *Random field modeling of elastic properties using homogenization*. Journal of Engineering Mechanics, 127(1):27–36, 2001.

[75] JANSON, S.: *Gaussian Hilbert Spaces*. Cambridge University Press, Cambridge, 1997.

[76] JARDAK, M.; C.-H. SU; and G. KARNIADAKIS: *Spectral polynomial chaos solutions of the stochastic advection equation*. SIAM Journal of Scientific Computing, 17(1-4):319–338, 2002.

[77] JEGGLE, H.-G.: *Nichtlineare Funktionalanalysis*. B.G. Teubner, Stuttgart, 1979.

[78] KAMIŃSKI, M.: *Stochastic finite element method homogenization of heat conduction problem in fiber composites*. Structural Engineering and Mechanics, 11(4):373–392, 2001.

[79] KAMIŃSKI, M. and M. KLEIBER: *Perturbation based stochastic finite element method for homogenization of two-phase elastic composites*. Computer and Structures, 78:811–826, 2000.

[80] KEESE, A.: *A review of recent developments in the numerical solution of stochastic PDEs (stochastic finite elements)*. Informatikbericht 2003-6, Technische Universität Braunschweig, Brunswick, `http://opus.tu-bs.de/opus/volltexte/2003/504/`, 2003.

[81] KEESE, A. and H. G. MATTHIES: *Efficient solvers for nonlinear stochastic problems*. In *Proceedings of the Fifth World Congress on Computational Mechanics, 7.-12. July*, Wien, `http://wccm.tuwien.ac.at/publications/Papers/fp81007.pdf`, 2002.

[82] KEESE, A. and H. G. MATTHIES: *Adaptivity and sensitivity for stochastic problems*. In P. SPANOS and G. DEODATIS (editors), *Computational Stochastic Mechanics 4*, 311–316, Millpress, 2003.

[83] KEESE, A. and H. G. MATTHIES: *Hierarchical parallel solution of stochastic systems*. In K. J. BATHE (editor), *Computational Fluid and Solid Mechanics 2003*, volume 2, 2023–2025, Elsevier, Amsterdam, 2003.

[84] KEESE, A. and H. G. MATTHIES: *Numerical methods and Smolyak quadrature for nonlinear stochastic partial differential equations*. submitted, 2003.

[85] KEESE, A. and H. G. MATTHIES: *Parallel computation of stochastic groundwater flow*. In *Proceedings of the NIC Symposium 2004*, Jülich, Germany, submitted, preprint at `http://opus.tu-bs.de/opus/volltexte/2003/505/`, 2003.

[86] KEESE, A. and H. G. MATTHIES: *Parallel solution of stochastic PDEs*. Proceedings in Applied Mathematics and Mechanics, 2:485–486, 2003.

[87] KEESE, A. and H. G. MATTHIES: *Sparse quadrature as an alternative to Monte Carlo for stochastic finite element techniques*. Submitted, 2003.

[88] KHURI, A. and J. CORNELL: *Response Surfaces: Designs and Analyses*. Dekker, New York, NY, 1987.

[89] KLEIBER, M.; H. ANTUNEZ; and T. D. HIEN: *Parameter Sensitivity in Nonlinear Mechanics*. John Wiley & Sons, Chichester, 1997.

[90] KLEIBER, M. and T. D. HIEN: *The Stochastic Finite Element Method. Basic Perturbation Technique and Computer Implementation*. John Wiley & Sons, Chichester, 1992.

[91] KLOEDEN, P. E. and E. PLATEN: *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin, 1995.

[92] KNUTH, D. E.: *The Art of Computer Programing, Vol 2: Seminumerical Algorithms*. Addison-Wesley, Reading, MA, 1981.

[93] KRÉE, P. and C. SOIZE: *Mathematics of Random Phenomena—Random vibrations of mechanical structures*. D. Reidel, Dordrecht, 1986.

[94] KUIREGHIAN, A. D. and J.-B. KE: *The stochastic finite element method in structural reliability*. Journal of Engineering Mechanics, 117(12):83–91, 1988.

[95] LEHOUCQ, R. B.; D. C. SORENSEN; and C. YANG: *ARPACK user's guide: solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, Philadelphia, PA, 1998.

[96] LI, R. and R. GHANEM: *Adaptive polynomial chaos expansions applied to statistics of extremes in nonlinear random vibration*. Probabilistic Engineering Mechanics, 13:125–136, 1998.

[97] LUCOR, D. and G. KARNIADAKIS: *Stochastic flow-structure interactions*. In K. J. BATHE (editor), *Computational Fluid and Solid Mechanics 2003*, volume 2, 1426–1429, Elsevier, Amsterdam, 2003.

[98] MADSEN, H. O.: *PROBAN: Theoretical manual for external release*. Technical report no. 88–2005, AS Veritas, Oslo, cited in [20], 1988.

[99] MAGLARAS, G.; E. NIKOLAIDIS; R. T. HAFKA; and H. H. CUDNEY: *Analytical-experimental comparison of probabilistic and fuzzy set based methods for designing under uncertainty*. In NATKE and BEN-HAIM [119], 1997.

[100] MALLIAVIN, P.: *Stochastic Analysis*. Springer, Berlin, 1997.

[101] MARCHUK, G. I.: *Adjoint Equations and Analysis of Complex Systems*. Kluwer, Dordrecht, 1995.

[102] MARTINEZ, W. L. and A. R. MARTINEZ: *Computational Statistics Handbook with Matlab*. Chapman & Hall, Boca Raton, FL, 2002.

[103] MASCAGNI ET AL., M.: *SPRNG 2.0, scalable parallel pseudo-random number generator library version 2.0*. Florida State University, Tallahassee, FL, `http://sprng.cs.fsu.edu/`, 1999.

[104] MASCHHOFF, K. and D. C. SORENSEN: *PARPACK: Parallel version of ARPACK for solving large scale eigenvalue problems*. Department of Computational and Applied Mathematics, Rice University, Houston, TX, `http://www.caam.rice.edu/~kristyn/parpack_home.html`, 1996.

[105] The MathWorks Inc, Natick, MA: *MATLAB External Interfaces API*. `http://www.mathworks.com/`, 1992.

[106] The MathWorks Inc, Natick, MA: *MATLAB User's Guide*. `http://www.mathworks.com/`, 1992.

[107] MATTHIES, H. G.; C. E. BRENNER; C. G. BUCHER; and C. G. SOARES: *Uncertainties in probabilistic numerical analysis of structures and solids—stochastic finite elements*. Structural Safety, 19(3):283–336, 1997.

[108] MATTHIES, H. G. and C. G. BUCHER: *Finite elements for stochastic media problems*. Computer Methods in Applied Mechanics and Engineering, 168:3–17, 1999.

[109] MATTHIES, H. G. and A. KEESE: *Multilevel methods for stochastic systems*. In *ECCM-2001, Proceedings of the Second European Conference on Computational Mechanics*, Cracow, Poland, 2001.

[110] MATTHIES, H. G. and A. KEESE: *Multilevel solvers for the analysis of stochastic systems*. In K. BATHE (editor), *Computational Fluid and Solid Mechanics*, 1620–1622, Elsevier, Amsterdam, 2001.

[111] MATTHIES, H. G. and A. KEESE: *Fast solvers for the white noise analysis of stochastic systems*. Proceedings in Applied Mathematics and Mechanics, 1:456–457, 2002.

[112] MATTHIES, H. G. and A. KEESE: *Fragen der numerischen Integration bei stochastischen finiten Elementen für nichtlineare Probleme*. In K. GÖRLEBECK; L. HEMPEL; and C. KÖNKE (editors), *Proceedings of the 16th international Conference on the Applications of Computer Science and Mathematics, Architecture and Civil Engineering*, preprint at `http://opus.tu-bs.de/opus/volltexte/2003/470/`, 2003.

[113] MATTHIES, H. G. and A. KEESE: *Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations*. Submitted, 2003.

[114] MATTHIES, H. G. and G. STRANG: *The solution of nonlinear finite element equations*. International Journal for Numerical Methods in Engineering, 14:1613–1626, 1979.

[115] MC CORMICK, S. (editor): *Multigrid Methods*. SIAM, Philadelphia, PA, 1986.

[116] MELCHERS, R. E.: *Structural Reliability Analysis and Prediction*. John Wiley & Sons, Chichester, 1999.

[117] Message Passing Interface Forum, University of Tennessee, Knoxville, Tennessee: *MPI: A Message Passing Standard*. `http://www.netlib.org/mpi`, 1995.

[118] MÖLLER, B.; W. GRAF; M. BEER; and J.-U. SICKERT: *Fuzzy stochastic finite element method*. In K. J. BATHE (editor), *Computational Fluid and Solid Mechanics 2003*, volume 2, 2074–2077, Elsevier, Amsterdam, 2003.

[119] NATKE, H. G. and Y. BEN-HAIM (editors): *Uncertainty: Models and Measures*. Akademie Verlag, Berlin, 1997.

[120] NEMAT-NASSER, S. and M. HORI: *Micromechanics: Overall Properties of Heterogeneous Materials*. North Holland, Amsterdam, 1993.

[121] NIEDERREITER, H.: *Random Number Generation and quasi-Monte Carlo Methods*. SIAM, Philadelphia, PA, 1992.

[122] NOVAK, E.: *Numerische Verfahren für hochdimensionale Probleme und der Fluch der Dimension*. Jahresbericht der DMV, 101:151–177, 1999.

[123] NOVAK, E. and K. RITTER: *High dimensional integration of smooth functions over cubes*. Numerische Mathematik, 75:79–97, 1996.

[124] NOVAK, E. and K. RITTER: *The curse of dimension and a universal method for numerical integration*. In G. NÜRNBERGER; J. W. SCHMIDT; and G. WALZ (editors), *Multivariate Approximation and Splines*, 177–188, Birkhäuser, Basel, 1997.

[125] NOVAK, E. and K. RITTER: *Simple cubature formulas with high polynomial exactness*. Constructive Approximation, 15:499–522, 1999.

[126] ODEN, J. T. and L. F. DEMKOWICZ: *Applied Functional Analysis*. CRC Press, Boca Raton, FL, 1996.

[127] ODEN, J. T. and J. N. REDDY: *An Introduction to the Mathematical Theory of Finite Elements*. John Wiley & Sons, Chichester, 1976.

[128] OGORODNIKOV, V. A. and S. M. PRIGARIN: *Numerical Modelling of Random Processes and Fields*. VSP, Utrecht, 1996.

[129] ØKSENDAL, B.: *Stochastic Differential Equations, An introduction with applications*. Springer, Berlin, 5^{th} edition, 1998.

[130] ORTEGA, J. M. and W. C. RHEINBOLDT: *Iterative solution of nonlinear equations in several variables*. SIAM, Philadelphia, PA, 2000.

[131] OSNES, H. and H. P. LANGTANGEN: *An efficient probabilistic finite element method for stochastic groundwater flow*. Advances in Water Resources, 22(2):185–195, 1998.

[132] PAPADRAKAKIS, M. and V. PAPADOPOULOS: *Robust and efficient methods for stochastic finite element analysis using Monte Carlo simulation*. Computer Methods in Applied Mechanics and Engineering, 134:325–340, 1996.

[133] PAPOULIS, A.: *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, Singapore, 3^{rd} edition, 1991.

[134] PARLETT, B. N.: *The symmetric Eigenvalue Problem*. SIAM, Philadelphia, PA, 1998.

[135] PELLISSETTI, M. and R. GHANEM: *Iterative solution of systems of linear equations arising in the context of stochastic finite elements*. Advances in Engineering Software, 31(8-9):607–616, 2000.

[136] PELLISSETTI, M. F.: *On Estimating the Error in Stochastic Model-Based Predictions*. Doctoral thesis, The Johns Hopkins University, Baltimore, MA, 2003.

[137] PETRAS, K.: *Fast calculation of coefficients in the Smolyak algorithm*. Numerical Algorithms, 26:93–109, 2001.

[138] PETRAS, K.: *SMOLPACK—a software for Smolyak quadrature with delayed Clenshaw-Curtis basis-sequence*. `http://www-public.tu-bs.de:8080/~petras/software.html`, 2002.

[139] PETRAS, K.: *Asymptotically minimal Smolyak cubature*. submitted, `http://citeseer.nj.nec.com/389404.html`, 2003.

[140] PIERCE, N. A. and M. B. GILES: *Adjoint recovery of superconvergent functionals from PDE approximations*. SIAM Review, 42(2):247–264, 2000.

[141] POTTHOFF, J.; G. VÅGE; and H. WATANABE: *Generalized solutions of linear parabolic stochastic partial differential equations*. Applied Mathematics and Optimization, 38:95–107, 1998.

[142] PRESS, W. H. and G. R. FARRAR: *Recursive stratified sampling for multidimensional Monte Carlo integration*. Computers in Physics, 4(2):190–195, 1990.

[143] PRESS, W. H.; S. A. TEUKOLSKY; W. T. VETTERLING; and B. P. FLANNERY: *Numerical Recipes in C—The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2nd edition, 1997.

[144] PRIGARIN, S. M.: *Spectral Models of Random Fields in Monte Carlo Methods*. VSP, Utrecht, 2001.

[145] RANNACHER, R.: *The dual-weighted-residual method for error control and mesh adaptation in finite element methods*. In J. R. WHITEMAN (editor), *MAFELAP X, Proceedings of the Tenth Conference on The Mathematics of Finite Elements and Applications*, 1999.

[146] RAO, S. S. and J. P. SAWYER: *Fuzzy finite element approch for the analysis of imprecisely defined systems*. AIAA Journal, 33:2364–2370, 1995.

[147] REED, M. and B. SIMON: *Methods of modern mathematical physics II: Fourier Analysis and Self-Adjointness*. Academic Press, New York, NY, 1975.

[148] REED, M. and B. SIMON: *Methods of modern mathematical Physics IV: Analysis of Operators*. Academic Press, New York, NY, 1978.

[149] REED, M. and B. SIMON: *Methods of modern mathematical Physics I: Functional Analysis*. Academic Press, New York, NY, 1980.

[150] RIPLEY, B. D.: *Statistical inference for spatial processes*. Cambridge University Press, Cambridge, 1988.

[151] ROSENBLATT, M.: *Remarks on a multivariate transformation*. The Annals of Mathematical Statistics, 23:470–472, cited in [63], 1952.

[152] ROZANOV, Y. A.: *Random Fields and Stochastic Partial Differential Equations*. Kluwer, Dordrecht, 1998.

[153] SAAD, Y.: *Numerical methods for large eigenvalue problems: Theory and algorithms*. Manchester University Press, Manchester, 1992.

[154] SCHIEKOFER, T. and G. ZUMBUSCH: *Software concepts of a sparse grid finite difference code*. In W. HACKBUSCH and G. WITTUM (editors), *Concepts of Numerical Software*, Vieweg, Braunschweig, 1998.

[155] SCHUËLLER, G. and P. SPANOS: *Monte Carlo Simulation*. Balkema, Rotterdam, 2001.

[156] SCHUËLLER, G. I.: *A state-of-the-art report on computational stochastic mechanics*. Probabilistic Engineering Mechanics, 14(4):197–321, 1997.

[157] SCHUËLLER, G. I.: *Recent developments in structural computational stochastic mechanics*. In B. TOPPING (editor), *Computational Mechanics for the Twenty-First Century*, 281–310, Saxe-Coburg Publications, Edingburgh, 2000.

[158] SCHULZ, V.; A. BARDOSSY; and R. HELMIG: *Conditional statistical inverse modeling in groundwater flow by multigrid methods*. Computational Geosciences, 3:49–68, 1999.

[159] SCHWAB, C. and R.-A. TODOR: *Sparse finite elements for elliptic problems with stochastic data*. Research Report No. 2002-05, ETH Zürich, Zürich, 2002.

[160] SCHWARZ, H. R.: *Numerische Mathematik*. B.G. Teubner, Stuttgart, 1993.

[161] SCHÜRER, R.: *A comparison between (quasi-)Monte Carlo and cubature rule based methods for solving high-dimensional integration problems.*. Mathematics and Computers in Simulation, in press, 2003.

[162] SMITH, P. L.: *A Primer for Sampling Solids, Liquids and Gases*. ASA and SIAM, Philadelphia, PA, 2001.

[163] SMOLYAK, S. A.: *Quadrature and interpolation formulas for tensor products of certain classes of functions*. Soviet Mathematics Dokl., 4:240–243, 1963.

[164] SOBCZYK, K. and D. J. KIRNER: *Stochastic Modeling of Microstructures*. Birkhäuser, Basel, 2001.

[165] STRANG, G. and G. J. FIX: *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, Cambridge, MA, 1988.

[166] STRANG, G. and H. MATTHIES: *Numerical computations in nonlinear mechanics*. In R. GLOWINSKI and J. L. LIONS (editors), *Computing Methods in Applied Sciences and Engineering; Proceedings of the Fourth International Symposium on Computing Methods in Applied Sciences and Engineering*, North Holland, Amsterdam, 1979.

[167] STROUSTRUP, B.: *The C++ Programming Language*. Addison-Wesley, Reading, MA, 3rd edition, 1997.

[168] STÜBEN, K.: Algebraic Multigrid (AMG): An Introduction with Applications, *GMD Report No. 70*. Technical report, Gesellschaft für Mathematik und Datenverarbeitung mbH, St. Augustin, 1999.

[169] SUDRET, B. and A. D. KIUREGHIAN: *Stochastic finite element methods and reliability. A state-of-the-art-report*. Technical Report UCB/SEMM-2000/08, University of California, Berkeley, CA, 2000.

[170] Sun Microsystems, Inc: *RFC 1014 - XDR: External Data Representation standard*. `http://www.faqs.org/rfcs/rfc1014.html`, 1987.

[171] THETING, T. G.: *Solving Wick-stochastic boundary value problems using a finite element method*. Stochastics and Stochastic Reports, 70(3–4):241–270, 2000.

[172] TORQUATO, S.: *Random Heterogeneous Materials*. Springer, Berlin, 2000.

[173] VAN TREES, H. L.: *Detection, Estimation and Modulation Theory, Part I*. John Wiley & Sons, Chichester, 1968.

[174] VANMARCKE, E.: *Random Fields: Analysis and Synthesis*. The MIT Press, Cambridge, MA, 3rd edition, 1988.

[175] WALSH, J. B.: *An introduction to stochastic partial differential equations*. In *École d'Été de Probabilités de Saint Flour XIV*, Springer, Berlin, 1984.

[176] WAUBKE, H.: *Dynamische Berechnungen für den Halbraum mit streuenden Parametern mittels orthogonaler Polynome*. Berichte aus dem konstruktiven Ingenieurbau 2/96, Lehrstuhl für Baumechanik, Technische Universität München, Munich, 1996.

[177] WERNER, D.: *Funktionalanalysis*. Springer, Berlin, 2^{nd} edition, 1997.

[178] WHITTLE, P.: *On stationary processes in the plane*. Biometrika, 41:434–449, 1954.

[179] WICK, G. C.: *The evaluation of the collinear matrix*. Physical Review Letters, 80:268–272, cited in [71], 1950.

[180] WIENER, N.: *The homogeneous chaos*. American Journal of Mathematics, 60:897–936, 1938.

[181] XIU, D. and G. E. KARNIADAKIS: *Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos*. Computer Methods in Applied Mechanics and Engineering, 191:4927–4948, 2002.

[182] XIU, D. and G. E. KARNIADAKIS: *The Wiener-Askey polynomial chaos for stochastic differential equations*. SIAM Journal of Scientific Computing, 24(2):619–644, 2002.

[183] XIU, D. and G. E. KARNIADAKIS: *Modeling uncertainty in flow simulations via generalized polynomial chaos*. Journal of Computational Physics, 187(1):137–167, 2003.

[184] XIU, D.; D. LUCOR; and G. KARNIADAKIS: *Modeling uncertainty in flow-structure interactions*. In K. J. BATHE (editor), *Computational Fluid and Solid Mechanics*, volume 2, 1420–1423, Elsevier, Amsterdam, 2001.

[185] XIU, D.; D. LUCOR; C.-H. SU; and G. E. KARNIADAKIS: *Stochastic modeling of flow-structure interactions using generalized polynomial chaos*. Journal of Fluid Engineering, 124:51–69, 2002.

[186] YAMAZAKI, F.; M. SHINOZUKA; and G. DASGUPTA: *Neumann expansion for stochastic finite element analysis*. American society of civil engineers (ASCE), Journal of Engineering Mechanics, 114(8):1335–1354, 1988.

[187] YU, Y.: *Coupling of ANSYS with a Stochastic Finite Element Solver and Visualisation of Results*. Master's thesis, Technische Universität Braunschweig, Institut für Wissenschaftliches Rechnen, Brunswick, 2003.

[188] YU, Y.; A. KEESE; and H. G. MATTHIES: *Coupling a stochastic finite element solver with ANSYS and visualization of the results*. In *Proceedings of the 21st CAD-FEM Users' Meeting 2003, International Congress on FEM Technology*, Potsdam, submitted, 2003.

[189] ZADEH, L. A.: *Fuzzy sets*. Information and Control, 8(3):338–353, 1965.

[190] ZIENKIEWICZ, O. C. and R. L. TAYLOR: *The Finite Element Method—Volume 1, the basis*. Butterwort-Heinemann, Oxford, 5^{th} edition, 2000.

[191] ZOHDI, T. I. and P. WRIGGERS: *Computational micro-macro material testing*. Archives of Computational Methods in Engineering, 8(2):131–228, 2001.

[192] ZUMBUSCH, G.: *A Sparse Grid PDE Solver; Discretization, Adaptivity, Software Design and Parallelization*. In H. P. LANGTANGEN; A. M. BRUASET; and E. QUAK (editors), *Advances in Software Tools for Scientific Computing (Proceedings SciTools '98)*, 133–177, Springer, Berlin, 2000.

Publications Written in the Course of This Thesis

Parts of the results of this thesis were already published or submitted for publication. For the reader's convenience, they are summarised and commented below.

Overview of Publications

First versions of the iterative solvers for linear SPDEs were presented in [109–111] and the first version of the solver for nonlinear SPDEs was described in [81]. Later versions of the nonlinear solver, a solver based on orthogonal projection, and techniques for high-dimensional integration were published in [84, 87, 112, 113].

An older and less efficient version of the parallel solver was presented in [83, 86]. A newer description of the parallel solver is [85]. First results on adaptive techniques based on dual methods were published in [82].

A review of the theory of SPDEs and an overview over various discretisation techniques for linear and nonlinear SPDEs was given in the review [80]. A unified presentation of the discretisation of various types of SPDEs was submitted for publication [113]. Aspects of the software framework and of visualisation techniques were described in the master's thesis [187] and were submitted for publication [188].

Commented List of Publications

[80] KEESE, A.: *A review of recent developments in the numerical solution of stochastic PDEs (stochastic finite elements)*. Informatikbericht 2003-6, Technische Universitat Braunschweig, Braunschweig, `http://opus.tu-bs.de/opus/volltexte/2003/504/`, 2003.
Comment: This report reviews stochastic field discretisations, the theory of SPDEs, discretisation methods, and numerical techniques for stochastic finite element techniques.

[81] KEESE, A. and H. G. MATTHIES: *Efficient solvers for nonlinear stochastic problems*. In *Proceedings of the Fifth World Congress on Computational Mechanics, 7.-12. July*, Wien, `http://wccm.tuwien.ac.at/publications/Papers/fp81007.pdf`, 2002.
Comment: First work on the solution of nonlinear SPDEs by approximate Newton and quasi-Newton methods; see Section 4.4.

[82] KEESE, A. and H. G. MATTHIES: *Adaptivity and sensitivity for stochastic problems*. In P. SPANOS and G. DEODATIS (editors), *Computational Stochastic Mechanics 4*, 311–316, Millpress, 2003.
Comment: This article describes the adaptive method presented in Section 4.5.

[83] KEESE, A. and H. G. MATTHIES: *Hierarchical parallel solution of stochastic systems*. In K. J. BATHE (editor), *Computational Fluid and Solid Mechanics 2003*, volume 2, 2023–2025, Elsevier, Amsterdam, 2003.
Comment: Here, multilevel methods for linear SPDEs are described; see Section 4.2.2.

[84] KEESE, A. and H. G. MATTHIES: *Numerical methods and Smolyak quadrature for nonlinear stochastic partial differential equations*. submitted, 2003.
Comment: This article introduces Smolyak quadrature for the solution of nonlinear SPDEs. It presents the orthogonal projection method of Section 3.4.2, it presents the theory for nonlinear SPDEs of Section 2.3.3, and it introduces the stochastic Galerkin method of Sections 3.4.3 and 4.4.

[85] KEESE, A. and H. G. MATTHIES: *Parallel computation of stochastic groundwater flow*. *Proceedings of the NIC Symposium 2004*, Jülich, submitted, `http://opus.tu-bs.de/opus/volltexte/2003/505/`, 2003.
Comment: This work presents an earlier version of the parallel solver of Section 4.3 and applies it to a stochastic groundwater flow example.

[86] KEESE, A. and H. G. MATTHIES: *Parallel solution of stochastic PDEs*. Proceedings in Applied Mathematics and Mechanics, 2:485–486, 2003.
Comment: Here, the first version of the parallel solver of Section 4.3 is presented.

[87] KEESE, A. and H. G. MATTHIES: *Sparse quadrature as an alternative to Monte Carlo for stochastic finite element techniques*. Submitted , 2003.
Comment: This article presents the direct integration technique of Section 3.3.

[109] MATTHIES, H. G. and A. KEESE: *Multilevel methods for stochastic systems*. In *ECCM-2001, Proceedings of the Second European Conference on Computational Mechanics*, Cracow, Poland, 2001.
Comment: This paper gives an account of the block-iterative methods and of the multilevel solvers for linear SPDEs of Section 4.2.

[110] MATTHIES, H. G. and A. KEESE: *Multilevel solvers for the analysis of stochastic systems*. In K. BATHE (editor), *Computational Fluid and Solid Mechanics*, 1620–1622, Elsevier, Amsterdam, 2001.
Comment: This work describes an early version of the multilevel methods for linear SPDEs of Section 4.2.

[111] MATTHIES, H. G. and A. KEESE: *Fast solvers for the white noise analysis of stochastic systems*. Proceedings in Applied Mathematics and Mechanics, 1:456–457, 2002.
Comment: Here, the block-iterative solvers and the multilevel methods for linear SPDEs of Section 4.2 are summarised.

[112] MATTHIES, H. G. and A. KEESE: *Fragen der numerischen Integration bei stochastischen finiten Elementen für nichtlineare Probleme*. In K. GÖRLEBECK; L. HEMPEL; and C. KÖNKE (editors), *Proceedings of the 16th international Conference on the Applications of Computer Science and Mathematics, Architecture and Civil Engineering*, preprint at `http://opus.tu-bs.de/opus/volltexte/2003/470/`, 2003.
Comment: This is a detailed presentation of the direct integration techniques of Section 3.3 in the context of Galerkin methods and of orthogonal projection techniques.

[113] MATTHIES, H. G. and A. KEESE: *Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations*. Submitted, 2003.
Comment: This paper gives a comprehensive account of the theory of SPDEs and of various discretisation techniques for SPDEs. New solution techniques are proposed that were not yet implemented.

[187] YU, Y.: *Coupling of ANSYS with a Stochastic Finite Element Solver and Visualisation of Results*. Master's thesis, Technische Universität Braunschweig, Institut für Wissenschaftliches Rechnen, Brunswick, 2003.
Comment: The coupling of StoFEL and ANSYS described in Section 5.2.2 and the visualisation methods used in Section 5.2.2 were implemented in this master's thesis.

[188] YU, Y.; A. KEESE; and H. G. MATTHIES: *Coupling a stochastic finite element solver with ANSYS and visualization of the results*. In *Proceedings of the 21st CAD-FEM Users' Meeting 2003, International Congress on FEM Technology*, Potsdam, 2003.
Comment: This is article about the coupling of StoFEL and ANSYS and about the visualisation techniques of Section 5.2.2.